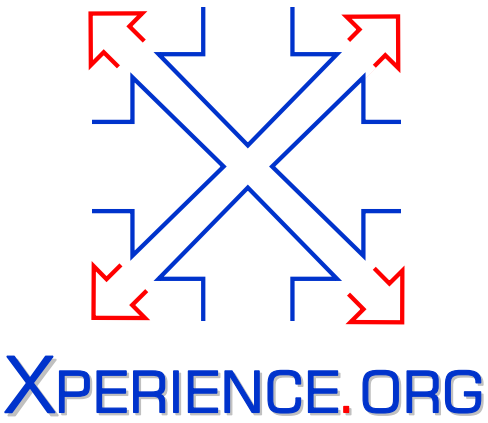
Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015



Deliverable Number:	D2.3.1
Deliverable Title:	Affordances and Categories (I): Report or scientific publication on exploration-based learning object affordances by means of semantic sensorimotor categories
Type (Internal, Restricted, Public):	PU
Authors:	D. Kraft, N. Krüger, S. Szedmak, J. Piater, F. Wörgötter, A. Ude, M. Steedman, T. Asfour
Contributing Partners:	SDU, UIBK, UEDIN, KIT, UGOE, JSI

Contractual Date of Delivery to the EC:	31-01-2012
Actual Date of Delivery to the EC:	06-02-2012

Contents

Chapter 1

Executive Summary

This deliverable deals with the work in WP 2.3 in year 1. After giving a brief outline of the term 'Object Action Complexes' (OACs), we discuss the relation of OACs and 'structural bootstrapping'. We define a number of concrete OACs which deliver the material for structural bootstrapping and describe some learning tasks with associated learning spaces in the sensorimotor and symbolic domain.

In section 2.1, we briefly present the concept of Object Action Complexes (OACs) which is the framework in which structural bootstrapping is taking place. OACs formalize adaptable sensorimotor schema grounded in real world experience. These then provide the data for structural bootstrapping processes as an outside-in process.

In section 2.2, we present concrete OACs that lead to the learning of object affordances. These cover

- explorative grasping with dexterous hands triggered by a deep hierarchical vision system,
- pushing behaviours in the context of object exploration and
- pushing in the context of learning by demonstration.

These OACs have been implemented on platforms at SDU, UGOE and JSI and are partly demonstrated in D5.2.1.

In section 2.3, we discuss the cross space (ECVxPMF) of an Early Cognitive Vision (ECV) representation and probabilistic manipulation functions (PMFs) as a basis for learning of sensory motor categories. This space covers a functional abstraction of the human visual system in terms of a deep hierarchy and a probabilistic action representation. In section 2.4 – 2.6, we give a number of examples of learning problems in the ECV space, in the space of PMFs and in the cross space ECVxPMF. The aim is to discover relevant substructures in the visual and motor domain as well as its cross space which can then be exploitet in inside-out processes to predict possible action affordances in novel contexts.

In the first year of Xperience we made first steps exploring these spaces by

- applying the concept of probabilistic manipulation functions to dexterous grasping,
- exploring the transition of grasping affordances across different contexts,
- applying the concept of probabilistic manipulation functions in the context of peg-in-hole actions,
- learning optimal kernels for generalization,
- investigating statistically feature grasp associations in still rather simple cases.

In section 2.7 we present work on learning action representations from change data.

To this deliverable 6 accepted papers are attached, 2 of the attached papers are in revision and 3 papers are in preparation for submission.

Chapter 2

Content of the Deliverable

In section 2.1, we briefly present the concept of Object Action Complexes and relate it to structural bootstrapping. In section 2.2, we present some concrete OACs and in section 2.3, we discuss the cross space (ECVxPMF) of an Early Cognitive Vision (ECV) representation and probabilistic manipulation functions (PMFs) as a basis for learning of sensory motor categories. In section 2.4 – 2.6 a number of examples of learning problems in the ECV space, in the space of PMFs and in the cross space ECVxPMF are given. Finally, in section 2.7 we present work on learning action representations from change data.

2.1 Object Action Complexes and Structural Bootstrapping

In this section, we outline two basic concepts relevant for Xperience. In section 2.1.1, we introduce the concept of Object Action Complexes (OACs) which has mainly been derived in the context of the PACO-PLUS project. In section 2.1.2 we then explain how OACs and structural bootstrapping relate to each other.

Attached is the paper [KGP+11] in which we give a formal definition of OACs. We put OACs in the context of structural bootstrapping in the attached paper [KPB+11]. In the attached paper [GKKed], we give a general review on the development of tool–use competences of infants as a basis for the modeling of structural bootstrapping processes. Text passages in the main part of this section are partly taken from these three attached papers.

2.1.1 Object Action Complexes

OACs formalize sensory-motor schemas [19, 5] and by that generate the required data for generalization. An OAC is a dynamic entity that gathers together the perceptions and associated actions involved in the performance of a habitual behaviour which is described in detail in the attached paper [KGP+11]. The schema represents knowledge generalised from all the experiences of the behaviour generated. It also includes knowledge about the context in which the behaviour was performed as well as expectations about the effects. During cognitive development, OACs are refined and combined.

An OAC's definition is split into three parts, (1) a *symbolic description* consisting of a prediction function defined over an attribute space, together with a measure of the reliability of the OAC, and (2) an *execution specification* that defines how the OAC is executed by the embodied system and generates experience in terms of 'experiments' and (3) a specification of how the learning associated with the OAC is realised based on the 'experiments' generated by the executed OACs. More formally (see also Fig. 2.1):

Definition 2.1.1. *An **Object-Action Complex (OAC)** is a triple*

$$(\mathtt{id}, T, M) \tag{2.1}$$

where:

- $\mathtt{id}$ *is a unique identifier for an execution specification,*

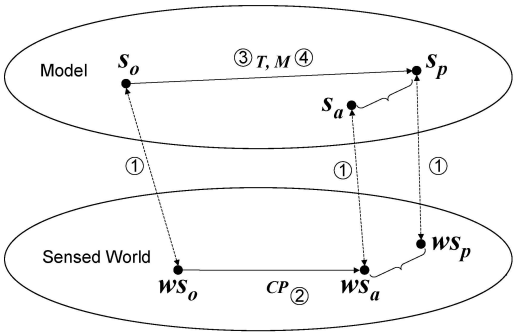


Figure 2.1: Graphical representation of an OAC and the OAC learning problems. This shows how the actual state ws_a (corresponding to s_a in the model) resulting from the execution of the control program CP diverges from the state s_p predicted by the OAC's prediction function T. This divergence drives the learning (i.e. refinement) of the OAC. For further explanation see text.

- $T : \mathcal{S} \rightarrow \mathcal{S}$ *is a prediction function defined on an attribute space* $\mathcal{S}$ *encoding the system's beliefs as to how the world (and the robot) will change if the control is executed, and*
- M *is a statistical measure representing the success of the OAC within a window over the past.*

An execution function `execute` can map an OAC `id` to an 'experiment' which is defined the following way:

Definition 2.1.2. *Given an attribute space* $\mathcal{S}$ *and an OAC with identifier* `id` *defined on* $\mathcal{S}$*, an* ***experiment*** *is a triple*

$$(s_o, s_p, s_a) \tag{2.2}$$

where:

- $s_o \in \mathcal{S}$*, captures the state of world before execution*
- $s_p \in \mathcal{S}$ *such that OAC* `id` *predicts that state* s_p *will result from its execution in* s_o*, i.e.,* $s_p = T_{id}(s_o)$*, and*
- $s_a \in \mathcal{S}$ *such that* s_a *is observed as a result of actually executing OAC* `id` *in state* s_o*.*

Thus, an experiment is an *empirical event* by which OACs will be grounded in sensory experience.

As an empirical grounded event, such experiments can be used to update OACs in cycles of execution and learning based on evaluations of their success. Note that sometimes an experiment is actually not used directly for learning but stored in some short term memory (see, e.g., [3]) until resources for learning are available (e.g., during 'sleeping phases').

The definition of OACs as capturing both symbolic and control knowledge about actions highlights a number of learning problems (addressing different aspects of the OAC as indicated in figure 2.1) that must be addressed for OACs to be effective (for details, see [KGP+11]). We note that while each of these learning problems can be addressed by recognising the differences between predicted states and those states actually achieved, they may still require different learning algorithms (e.g., Bayesian, neural network-like, parametric, non-parametric, etc.). It is up to the OAC designer to choose an appropriate learning mechanism.

2.1.2 Structural Bootstrapping

Cognitive development proceeds at two different levels of abstraction. Fig. 2.2 shows two such parallel tracks of development. On the bottom is the sensorimotor track which shows the development of sensorimotor schemas or OACs, which are observable in infant behaviour. Each node in the lower track corresponds to one OAC. A directed edge travels from each ancestor node to its descendents; for example

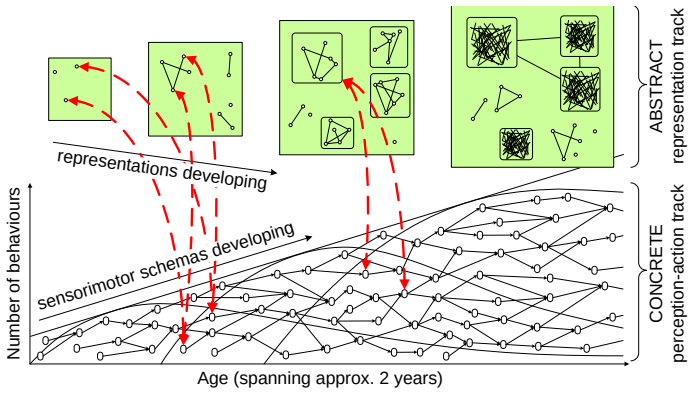


Figure 2.2: Conceptual diagram, overviewing infant developments on both a low level sensorimotor track and a higher level representational track; for explanation see text.

the OAC for pulling a string descends from a basic grasping OAC (a number of examples are outlined in [GKKed]). Some OACs have more than one ancestor. The top of Fig. 2.2 is the abstract track which shows the parallel development of the underlying world knowledge. Nodes in the upper track correspond to (for example) fragments of object knowledge which are common to a number of OACs, and fragments of spatial relationships; these are gradually linked up as development progresses (to the right), to eventually form a more comprehensive world knowledge.

Early fragments of object and spatial knowledge are likely to be very context specific, and are strongly associated with the OACs they have been abstracted from. It is only after a long developmental process (moving to the right in Fig. 2.2) that these fragments become more objective, and this developmental process must involve some sort of "representational redescription" [4].

For the lower track we see a developmental process in which a small set of innately defined OACs lead to a large variety of OACs through branching and specialisation. During this developmental process, the effects of the OAC become increasingly predictable and can then be used more and more purposefully by the cognitive agent for the planning of behaviour. In parallel to (and triggered by) the development of individual OACs, more generic world knowledge is built up; as illustrated in the upper track of Fig. 2.2. This is done through the abstraction of empirical data gained during the execution of the OACs.

There is a parallel development and interaction of observable behaviours and the increasing abstract world knowledge which is based on the experiments generated by the OACs. *Structural bootstrapping* addresses the process indicated by the red arrows (see figure 2.2) in which generated abstracted world knowledge effects the behavioral track by means of establishing new OACs or modulating existing OACs: More specifically, the dashed red arrows in Figure 2.2 illustrate bidirectional links between the abstract and sensorimotor tracks. To avoid clutter only six links are shown, but in reality all representational fragments will be linked to the OACs. In one direction (outside-in) representations are linked to the OAC they have been generalised from (and hence can immediately link to actions which can manipulate the represented object or spatial relation). In the other direction (inside-out), more advanced OACs make use of the newly formed representations, for example in their description of the context in which a behaviour may be performed, or its effects, or the control policy followed during execution of the schema. *These feedback processes are at the core of structural bootstrapping since they allow to apply abstracted concepts generated in the outside-in process as an inside-out process facilitating cognitive behaviour based on predictions derived from internal concepts.*

2.2 Learning object affordances by means of semantic sensory–motor categories

In the context of task 2.3.1, we introduce OACs in which exploration is triggered by initially predefined behaviours ([20]) and two OACs in the context of learning by demonstration ([ADTW11], [USSM12]). In [20] we provide an extended grasping reflex (as already introduced in D2.1.1) in the context of the Early Cognitive Vision system [17]. The attached paper [USSM12] describes a pushing OAC in the

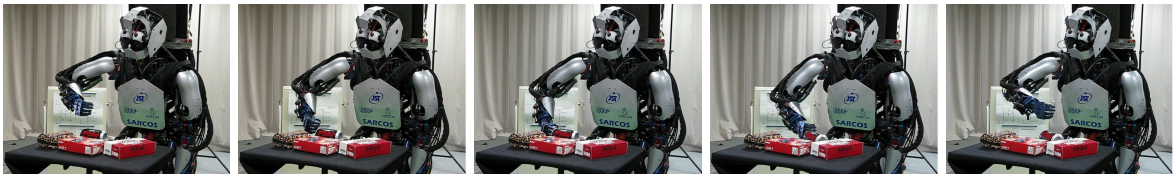


Figure 2.3: Humanoid robot performing the pushing to built up object representations

context of exploration while the attached paper [ADTW11] describes pushing in the context of learning by demonstration.

Explorative Learning by Pushing: Manipulation abilities of embodied agents enable them to learn object representations and their affordances by active exploration. In the first year of the project, we studied how a humanoid robot can exploit skills such as pushing to acquire new object knowledge (see figure 2.3). Our premise was that if object manipulability is taken into account, it is much easier to define the concept of an object than if only visual characteristics are used. We realized a system that uses a reflex pushing behaviour to detect and learn about objects that afford pushing (where "object affords pushing" is defined as "object moves when the robot attempts to push it").

The pushing behavior was learned by coaching, more specifically by kinesthetic guiding. For this purpose, the robot arm was guided through a number of configurations in space. Besides the demonstrated robot arm configurations, we also stored the associated 3-D Cartesian space positions as estimated by the robot's active vision. This knowledge is sufficient to generate pushing movements using statistical generalization and discrete dynamic systems. 3-D information extracted from the scene is used to generate initial hypotheses about existence of the objects. These hypotheses are used to trigger reflex pushes and the resulting image motion is used to either confirm or reject object hypotheses. We have shown in real-world experiments that in this way we can fully explore previously unknown objects and train state of the art classifiers that can be used for recognition. This work is described in the attached paper [USSM12].

Pushing in the context of learning by demonstration: This OAC deals with execution of a pushing action by means of Semantic Event Chains (SECs). SECs are generic schemes for manipulations and they basically encode object-action relations in spatiotemporal domain for further semantic analyses (e.g. classification). Earlier we have shown that SECs can be used for both classifying actions and categorizing objects considering their roles in actions [2]. Therefore, SECs are an important step in the context of Object-Action Complexes (OACs). In this deliverable we aim at explaining how a SEC can be enriched with motion information and how a (dual-object) pushing action can be executed directly from SECs. This work was published in the attached paper [ADTW11].

SECs provide decisive temporal anchor points which define critical relational changes between object pairs during the manipulation. In order to produce a SEC, we first segment the image sequence based on color and depth cues and then represent each frame by a graph, nodes and edges of which represent segments and their neighborhood relations, respectively[1]. Considering the topological changes (absence and/or occurrence of new nodes/edges) in graph structures, we calculate decisive temporal anchor points of the action. Movement segments between the temporal anchor points can be interpreted as meaningful movement primitives. Fig. 2.4 shows the SEC as well as decisive temporal anchor points with original frames, segments, and graphs of a sample pushing action performed by a 6 DOF Neuronics Katana robot arm in the Webots simulation environment [16]. The robot arm is pushing a red box to a green box until they touch each other. Each frame given in Fig. 2.4 corresponds to one column in the SEC, rows of which hold relational changes (e.g. (T)ouching, (N)on-touching, (A)bsence) between object pairs (e.g. red box and green box). The pushing action is encoded by a 3×7 matrix.

In order to be able to execute the action from its SEC, in addition to symbolic information we record motion information. For the pushing action start and endpoints of movements are important and are recorded at each decisive temporal anchor point. Movement primitives in this pushing example are defined as a motion vector from start point to the end point. The recorded start and end points defining the motion vectors are stored in association with the SEC. To make the agent learn a SEC-model with additional motion data we demonstrate the action 10 times. Fig. 2.5 illustrates learned motion vectors of the robot arm for the demonstrated pushing action. Dots represent points provided by the training

[1]Note that - in contrast to the ECV system described in section 2.3.1 – SECs give a high level description of the scene. In the cause of the Xperience projects we expect a tighter interaction of the ECV system and the SEC representation in which the ECV system will support the lower levels of the SEC approach.

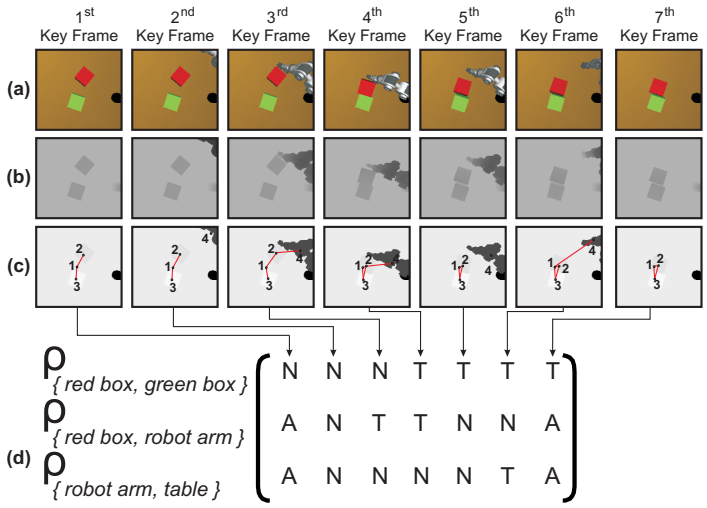


Figure 2.4: Semantic event chain representation. **(a)** Original *"Key Frames"*. **(b)** Corresponding depth maps. **(c)** Corresponding HSV color based segmented images with extracted main graphs. **(d)** Corresponding semantic event chain, which is a sequence-table, where each entry encodes the spatial relations between each segment pair at each main graph. T means that segments touch (denoted by red edges), N means that there is no edge between two segments, and absence of a previously existing segment yields A.

data set and starred points are the average locations. When executing the learned action, the robot manipulator will travels along the path described by motion vectors V_1, V_2, V_3 shown in the figure. Those motion vectors are crucial and provide enough information to execute the pushing action. The approach can also be extended to more general manipulation tasks, e.g. "pick & place" or "cutting a carrot with a knife". For more details see attached paper [ADTW11].

2.3 Semantic visual scene representations and probabilistic action representations

The OACs introduced in section 2.2 deliver the material in terms of experiments learning and structural bootstrapping is based on. The groups SDU and UIBK (and potentially also other groups) make use of a special space for learning (called ECVxPMF) which is introduced here. In section 2.3.1 (backrefering to D.2.1.1), we introduce the Early Cognitive Vision (ECV) system. In section 2.3.2, we present the concept of probabilistic manipulation functions (PMFs) and in section 2.3.3, we discuss the advantageous properties of the cross space of ECV and PMF (ECVxPMF) for a variety of learning tasks.

2.3.1 Early Cognitive Vision (ECV) system

The Early Cognitive Vision (ECV) system – as introduced in [17], [21] and deliverable D2.1.1 – is a hierarchically organized visual representation providing rich information on different levels of granularity. It represents appearance and geometric information as well as 2D and 3D information in a separate and explicit format for different kinds of visual entities (i.e., contours, junctions, texture and homogeneous areas). This system is a functional model of the human visual system and visual information can be processed close to real-time based on GPU technology (see, e.g., [18]). Such a representation allows for choosing an appropriate level of information for a certain task as outlined in [17]. This covers the use of 2D and 3D information as well as an appropriate level of granularity (for details see D3.1.1).

2.3.2 Learning Probabilistic Manipulation Functions

Objects can be grasped and manipulated in many ways. Human dexterity and robustness in handling objects is largely due to the human flexibility of adaptively choosing appropriate grasp configurations. As

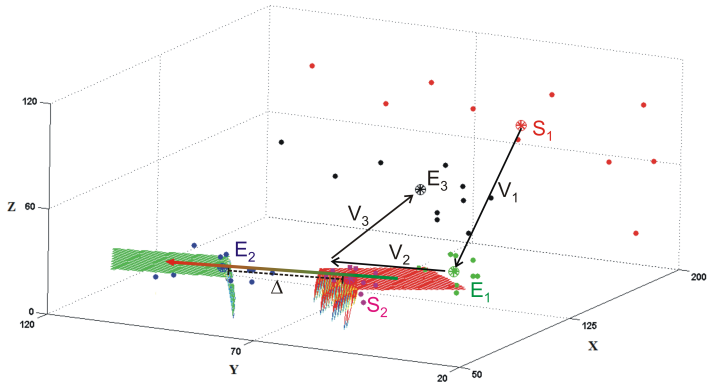


Figure 2.5: Start (S_1, S_2) and end (E_1, E_2, E_3) point distribution as provided by the training dataset. Starred points are the average locations. Robot manipulator travels along the path given by vectors V_1, V_2, V_3. The coordinate origin is at the center of the red object. The motion vector for pushing is depicted by the color-changing vector that connects red with green object. Distance Δ is defines as $|E_2 - S_2|$ and vector V_2 has length Δ.

a step towards this end, the concept of Probabilistic Manipulation Functions (PMFs) – developed as a co-operation between UIBK and SDU (in which SDUs main focus is on the application of the PMF concept in a variety of contexts) – seeks to represent the utility (success, robustness, ...) of a manipulation as a continuous function of grasp parameters. This offers the robot great flexibility in choosing and refining grasps under task constraints. As a first instance of this paradigm, we recently described the concept of *grasp densities* that model the spatial empirical distribution of successful grasps in object-relative pose space [6].

Grasp densities are learned from exploration by drawing grasps from a known pose-space distribution, performing them, and retaining successful grasps as importance-weighted particles of a nonparametric representation of the target grasp density. From such grasp (success) densities and the corresponding, analogous failure densities, posterior grasp success probabilities can in principle be estimated using Bayes' theorem (see [6]). Other recent work explores learning such success posteriors directly by regression [13].

The Xperience project seeks to further develop PGFs in various ways, including more efficient use of grasping experience by improving prediction and generalization capabilities, and going beyond grasping towards manipulative actions.

2.3.3 Learning in ECVxPMF

The two representations, on the vision side the hierarchical ECV system outlined in section 2.3.1 and the probabilistic manipulation function as outlined in section 2.3.2, span a space in which efficient learning of actions and behaviors can take place. In particular important for facilitating the learning tasks relevant in the Xperience context are the following properties:

The view point invariant representations of the ECV space reduce the complexity of a number of learning problems: The problem of learning manipulation affordances from visual cues involves a number of problems. Learning manipulation affordances from 2D information faces the problem that a huge variation of visual input needs to be related to the same affordance requiring the covering of this space in the learning phase. In contrast, the view point invariant representations in terms of geometric and appearance relations in the ECV system reduce this variation by means of a more complex feature processing. The same argument holds for the learning of suitable representations for object recognition and pose estimation (see, e.g., D2.1.1).

The explicit coding of different aspects of visual information in the ECV space allows for making use of only specific aspects for the learning of specific tasks: Human have to solve multiple tasks (navigation, recognition, map-building, manipulation, etc.) at the same time. This makes a general purpose representation (such as the ECV system) advantageous (in contrast to, e.g., frogs which have fly detectors in the retina [14]) which in humans cover more than 2/3 of the visual cortex. For a specific task only certain aspects of this information are relevant. The

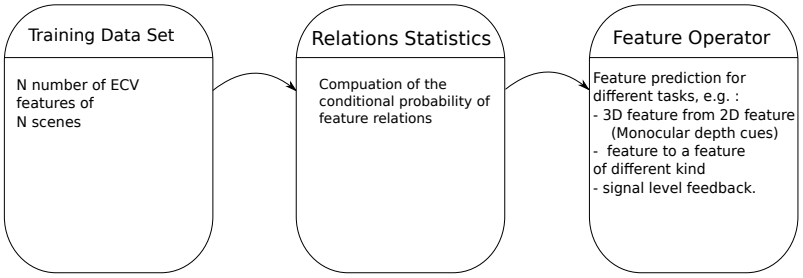


Figure 2.6: The Framework

explicit coding of this information makes these aspects separable (in contrast to, e.g., image feature histogram approaches such as SIFT [15]), allowing for efficient learning.

The hierarchical organization allows to address the learning problem at an appropriate level of granularity: Local features and their relations will quickly span a space that is so large that learning is hardly possible due to the combinatorics. The hierarchical representation allows for choosing a level of spatial extent that is appropriate for the specific learning problem.

The concept of Probabilistic Manipulation Functions allows for representing action experience on different objects in a way that can be directly related to the ECV space: PMFs are generally expressed in an object-centric reference frame. Thus, for rigid objects, they are rigidly attached to the object representation. Hierarchical and part-based representations such as the ECV system allow the association of PGFs at various levels, from an entire object down to individual ECV features. Object-level PMFs describe how an object can be grasped and can in principle be learned to a perfection limited only by the uncertainties inherent in perception and manipulation, but they apply to the given object only. Feature-level PMFs apply wherever the feature is found, but low-level features are not generally expressive enough to allow for learning of predictive PMFs. Good practical compromises can likely be found at intermediate levels of feature complexity that are sufficiently distinctive to be predictive of specific grasp configurations but sufficiently general to be found on a variety of objects.

The addressing of different learning problems within the same visual and action representation facilitates generalization across tasks: The transfer of experience across tasks is facilitated when different tasks are performed in the same visual and action representation. In this way, e.g., any knowledge on the computation of the object category can directly effect the knowledge on the manipulation affordances of objects: For example, assume a cup having a certain shape sub-structure 'handle' (object categorization) that can be used to lift it safely (object manipulation). Since the handle representation in both tasks share the same features it can then be reasoned on similar structures at other objects and their associated categories and affordances. In other words, knowledge generalisation – crucial for any cognitive agent – is facilitated in deep hierarchal structures.

2.4 Learning in ECV: Learning of interactions of entities based on statistical inference

Feature extraction in ECV involves operations that do local reasoning (i.e., at a pixel level). This local reasoning, in some cases, fails to extract features that are more likely to be extracted through a global reasoning. The need for a global perspective comes also in line with Gestalt theory in psychology [7].

The aim is to fulfill the above concepts through a learning framework within the ECV system. The framework will train the system on making task-specific predictions (operations) based on the statistical properties of inter-entity relations (e.g., the conditional probability of the angle between two entities). Which relations to use and how to compute their statistics can vary from task to task. Figure 2.6 shows the proposed framework and the interactions of its different components. At the beginning, the framework

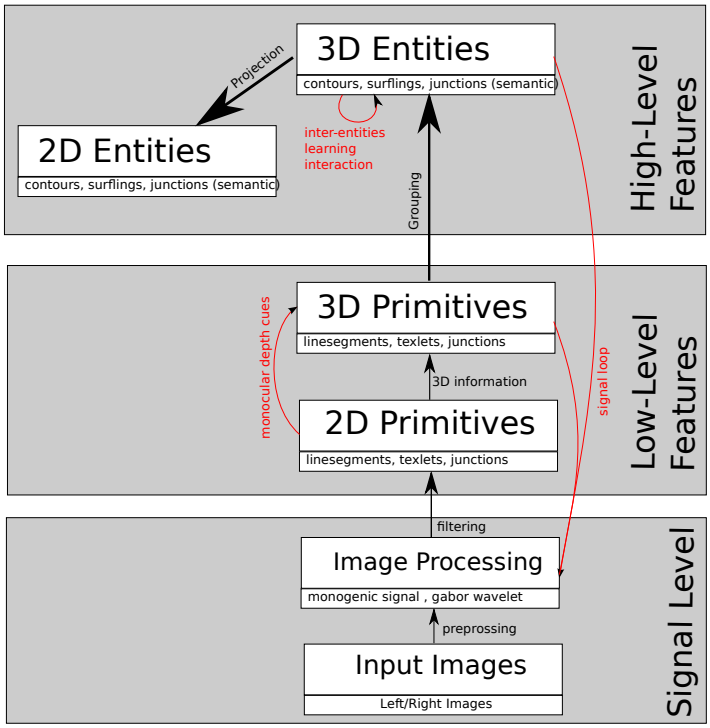


Figure 2.7: The ECV Hierarchy

needs to establish a large dataset of ECV features. From these relational statistics are computed. Finally, we will define feature operators which make use of the gathered statistics to perform operations on the input data. This is an example of structural bootstrapping in which abstracted world knowledge is gathered in terms of relational statistics to influence lower-level vision processes.

Examples for feature operations are (1) obtaining a 3D feature from a 2D feature (i.e., monocular depth cues) , (2) predicting features from features of different kinds (for completion and disambiguation) and (3) feed-backing the signal-level feature extraction process (to steer the extraction process by a global reasoning). During the first year we made first steps addressing item (2).

Depending on the operation, different ECV features will be required to interact with each other. This interaction may also involve entities at different levels of the ECV hierarchy. Figure 2.7 shows the possible ways of interaction within the hierarchy. The black arrows shows the normal flow of operations while the red arrows shows possible operations within the proposed framework.

2.5 Learning of PMFs

This section presents current work on theory and applications of PMFs as introduced in Sec. 2.3.2. Section 2.5.1 addresses the approximation of PMFs from samples, section 2.5.2 addresses the computation of grasp densities through simulation, and section 2.5.3 addresses the extension of probabilistic grasp functions to manipulation grasp functions at the example of peg-in-whole actions with flexible objects.

2.5.1 Data-efficient approximation of PMFs with kernel methods

To a first approximation, the prediction of the success or failure of a grasping action can be formulated as a binary classification problem. This classification can be built upon a set of tuples containing the results and the parameters of earlier experiments. These tuples consists of the indicator of the success and the object-relative position and orientation of the gripper. The objective of the learning task is then

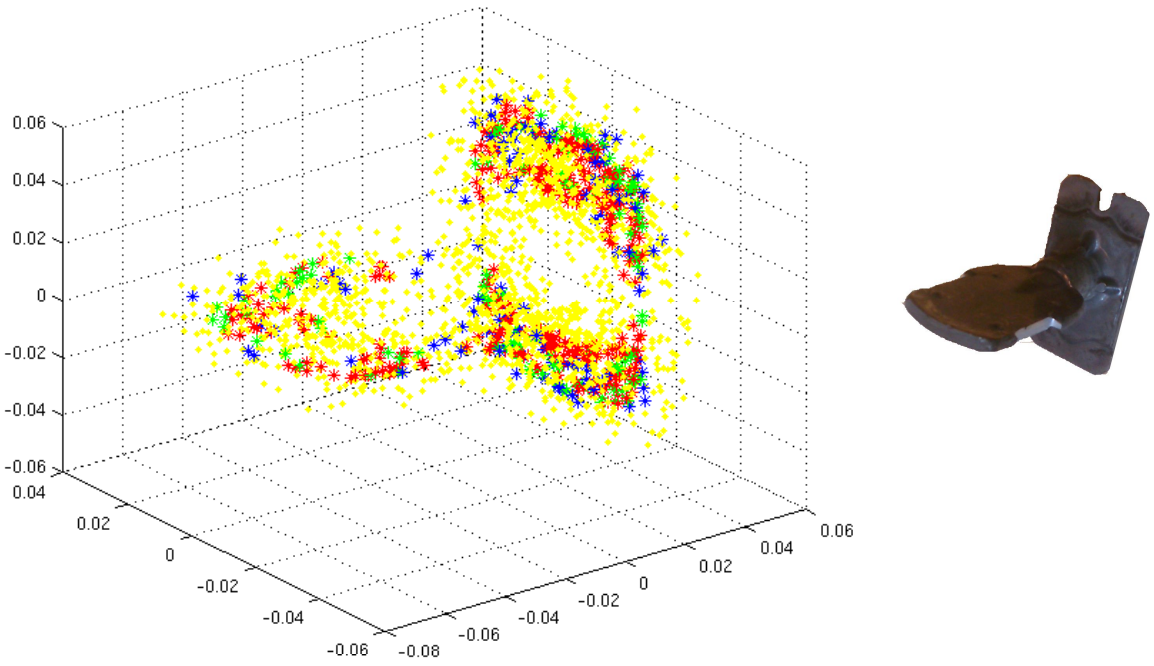


Figure 2.8: Distribution of the prediction of the successful grasping in the position space, $\mathbb{R}^3$. The target object resembles the illustration on the right. Legend: green = true positive, red = false positive, blue= false negative, yellow = true negative.

to predict the success in positions and orientations not observed in any experiment. To this end we need to approximate the probability distribution of success of the grasping for all possible cases based on the observed ones.

Our approach (described in detail in the attached draft [SKJP12]) is built upon a probability density function based representation of the positions, orientations and the grasping results. This approach assumes that the outcome of the grasping in an arbitrary position (orientation) with high probability follows the results of those observed positions (orientations) situated nearby. To implement this idea we apply a kernel-based approach, a generalization of the Support Vector Machine, where not only the input data (positions and orientations) but also the output (success indicators) are represented by kernels. The elements of these kernels are the value of inner products computed between the corresponding density functions which are localized in the observed values of the input and output parameters. In this setting, a point further away from an observed one has smaller probability of following the success of that observation. In his way, the distribution of the success is a weighted mixture of the distributions localized in the observations, where the weights are computed by a maximum margin based optimization model. Indicative results are shown in Fig. 2.8.

2.5.2 Probabilistic dexterous grasp function computed by dynamic simulation

Currently complete grasp affordences of objects have been computed based on specific environments with very few geometric constraints, e.g., grasp attempts are simulated in a free floating environment without any obstacles. These grasp affordances describe the likelihood of success of a 6D pose of the gripper relative to the object. When the environment is changed, new simulations are required to compute the new grasp affordances. This approach is very time consuming and when considering the possible combinations of environments, simply infeasible.

Reusing simulations from a free floating environment is tempting (see figure 2.9. However, when considering the new constraints on an object (friction with other obstacles, blocked path, etc.) in a more complex

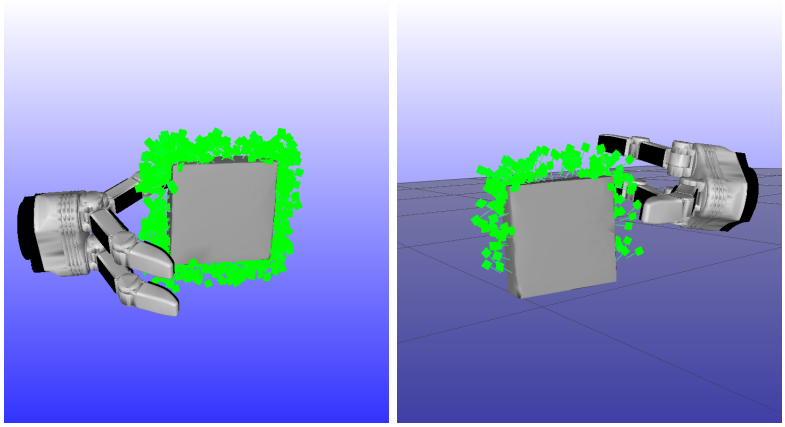


Figure 2.9: The left image show successfull grasps generated in an free floating environment without any obstacles. The right image show successfull grasps generated in an environment with gravity and a floor on which the object is resting.

environment it is not trivial to argue that successful grasps in the new environment is also successful in the lesser constrained environment. In [JKP+12] we present complete sets of grasp affordances for several objects in several different environments which enable us to quantitatively evaluate the resemblance of grasp affordances in different environments. We investigate the feasibility of reusing simulations run on lesser constrained environments and present a method to efficiently recompute grasp affordances in new environments without generating new simulations.

2.5.3 PiH with flexible objects

In [BFW+12] the concept of action learning based on kernel density estimation (see section 2.3.2) has been extended to peg-in-hole actions with flexible objects. The shape of an undeformed object is detected using a Kinect camera as in [11]. The depth information is subsequently used to create an triangular mesh model of the object. Until now rather regularly shaped objects have been used, so that the Bernoulli-Euler beam theory could be applied to model the deformation of the object due to gravity. Thereby the computational burden is still feasible, which would not be the case when more complex models based on finite-elements would be used.

In order use Kernel Density Estimation (KDE) for action learning, the action has been formulated such that the trajectory which the robot has to perform is defined by Bézier curve based on three points: a start point, one control point and the end point. The start point P_0 is defined such that the free end of the object is in front of the hole, the end point P_1 is chosen such that the gripper is in front of the hole with the object being in inserted. The remaining control point defines the trajectory from start to end point and will be learned. Figure 2.10 illustrates the trajectory. Currently it has been found to be sufficient to let the trajectory be defined by two-dimensional translations and one-dimensional rotation such that the trajectory is in $\mathbb{R}^2 \times SO(2)$.

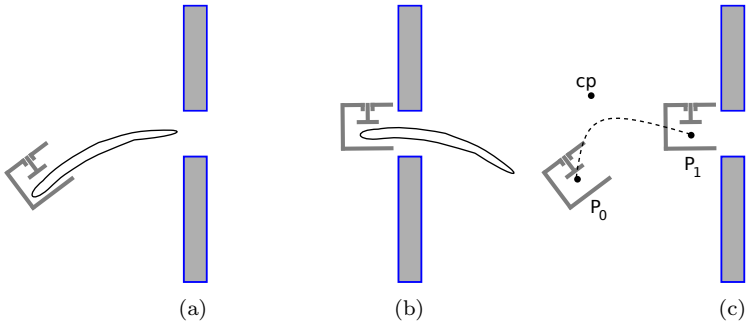


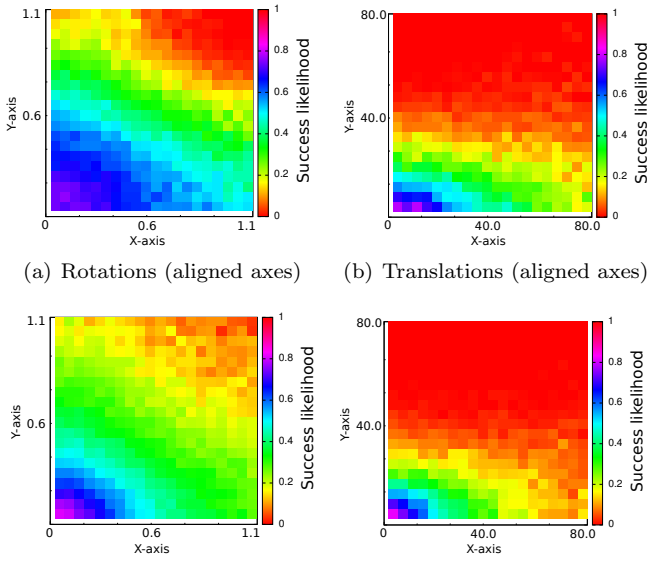
Figure 2.10: Illustration of (a) starting configuration P_0 and (b) target configuration P_1 for the peg-in-hole action. (c) shows a projection of the 3D trajectory based on P_0, P_1 and the control point cp.

2.6 Learning in ECV x PMF

In this section, we describe two works on structural bootstrapping learning . In the first example (section 2.6.1), statistics across a variety of objects is gathered to arrive at statistically motivated kernels in the context of grasp probability functions (see section 2.3.2) as well as first steps towards the learning of generic grasp affordances as described in section 2.6.2 and 2.6.3.

2.6.1 Meta Learning: Learning kernels by statistics

In the concept of grasp densities [6], kernel density estimation is used based on a six-dimensional kernel representing grasps with given position and orientation. For this so far an isotropic kernel has been used which exact shape has only been weakly justified. Instead in [BDPK11], we use an anisotropic kernel that is statistically based on measured conditional probabilities representing grasp success in the neighborhood of a successful grasp. The anisotropy has been determined utilizing a simulation environment that allowed for evaluation of large scale experiments (results for one object are shown on figure 2.11). The anisotropic kernel has been fitted to the conditional probabilities obtained from the experiments. Hereby it has been detected that it is advantageous to align the grasping action with the local structure. This becomes also explicit on figure 2.11 as the anisotropies are significantly pronounced when the actions has been aligned (top row). We then show that convergence is an important problem associated with the grasp density approach and we propose a measure for the convergence of the densities. In this context, we show that the use of the statistically grounded anisotropic kernels leads to a significantly faster convergence of grasp densities.



(a) Rotations (aligned axes) (b) Translations (aligned axes)

(c) Rotations (original axes) (d) Translations (original axes)

Figure 2.11: Histograms for the cone-object shows the success likelihood for grasping actions in the neighborhood of successful actions. Translations and rotations are shown on separate histograms. The top row shows the results when the axis of displacement have been aligned with the local edge, in the bottom row no alignment has been done.

2.6.2 Learning grasping/manipulation affordances

Our task is to connect grasping affordances to target objects. We need to find proper grasping approaches for given objects, and also to learn the classes of targets which can be grasped in a similar way. To this end we need to find appropriate feature representations of both the grasping affordances and the targets.

The grasping affordance and the shape of the targets can be characterized by similar models. Both types of models can be described by sets of points equipped with a proper internal structure. The points correspond to observations, e.g. grasping experiments, and the structure represents the interdependencies between these points for example the shape of the target objects.

The learning problem which can connect the affordances to the targets requires feature representations capable of capturing the real-world complexity of structures, e.g. shapes with holes, convex and concave substructures. To this end we apply so-called functional features which are sufficiently flexible to capture complex relations. In this approach the points correspond to functions, e.g. probability densities, and the structures are expressed in Hilbert spaces comprising these functions. In this way we can apply inner products, distances between these functions. Therefore we can create a complete geometry around the complex objects and affordances.

The learning model is based on the maximum-margin principle, a generalization of the well-known Support Vector Machine. Our model represents both the inputs and the outputs via kernels. Thus, relationships between very complex-structured objects can be learned.

The general method is described in a technical report as part of Deliverable D2.1.1 [22] in the context of shape learning. The same principles apply to represent the shape of probabilistic manipulation functions (PMFs). Future work aims to couple shape and PMF representations in order to infer one from the other.

2.6.3 ECV feature to grasp associations

In the attached paper [Tho12] an investigation of the feature to grasp association space in terms of the ECV (Early Cognitive Vision) high level features of 3D contours and 3D surfaces for single features. The investigation involves grasp simulation utilizing the robotic framework of RobWork and feature extraction relying in the ECV system (see figure 2.12). Simulations were performed with a two finger jaw gripper in a number of scenes with features of interest. A variety of situations with 3D contours and 3D surfaces has been designed and evaluated in order to expose the feature to grasp space (see figure 2.13). All the situations is compared against the ground truth feature reference frame and a ECV feature extracted reference frame showing that the surface extraction in particular seems promising. A qualitative analysis has been performed on the different situations through explanation of the acquired results (see figure 2.13) and finally a discussion of the results and future work in terms of ECV feature to grasp associations is carried out.

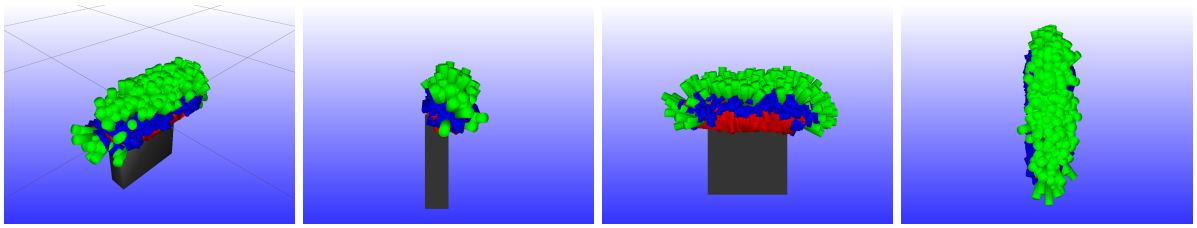


Figure 2.12: Visualization with a gripper in RobWork of successful grasps for a 3D contour situation.

2.7 Learning Action Representations from Change Data

Task 2.3.2 'Learning high–level action descriptions' is addressed in this subsection referring to the submitted paper [MZPS]. We focus on the problem of autonomous learning of relational action models from experience in the world. Existing techniques can learn STRIPS [8] action models in noiseless, fully observable worlds, but the problem is more challenging when observations of world state may be noisy and incomplete. Here we present a method which firstly can learn preconditions and effects of STRIPS actions in noisy, partially observable worlds, and secondly, can generate STRIPS action models suitable for use by automated planning systems. We assume the agent has already learnt to identify objects, and has acquired predicates to describe object attributes and relations between objects. Also we assume that the agent has previously acquired primitive actions and their immediate arguments. For example, it may have learnt to grasp an object. The specific sequence of motor actions to effect the grasp has a unique identifier, and it is known that the action requires an argument, namely the object to be grasped. What is unknown is the set of preconditions for the grasp action (e.g., an object is not already being grasped), and the effects (e.g., any object under the grasped object is now clear). Experience in the world is then developed through observing changes to object attributes and relations when motor-babbling with primitive actions.

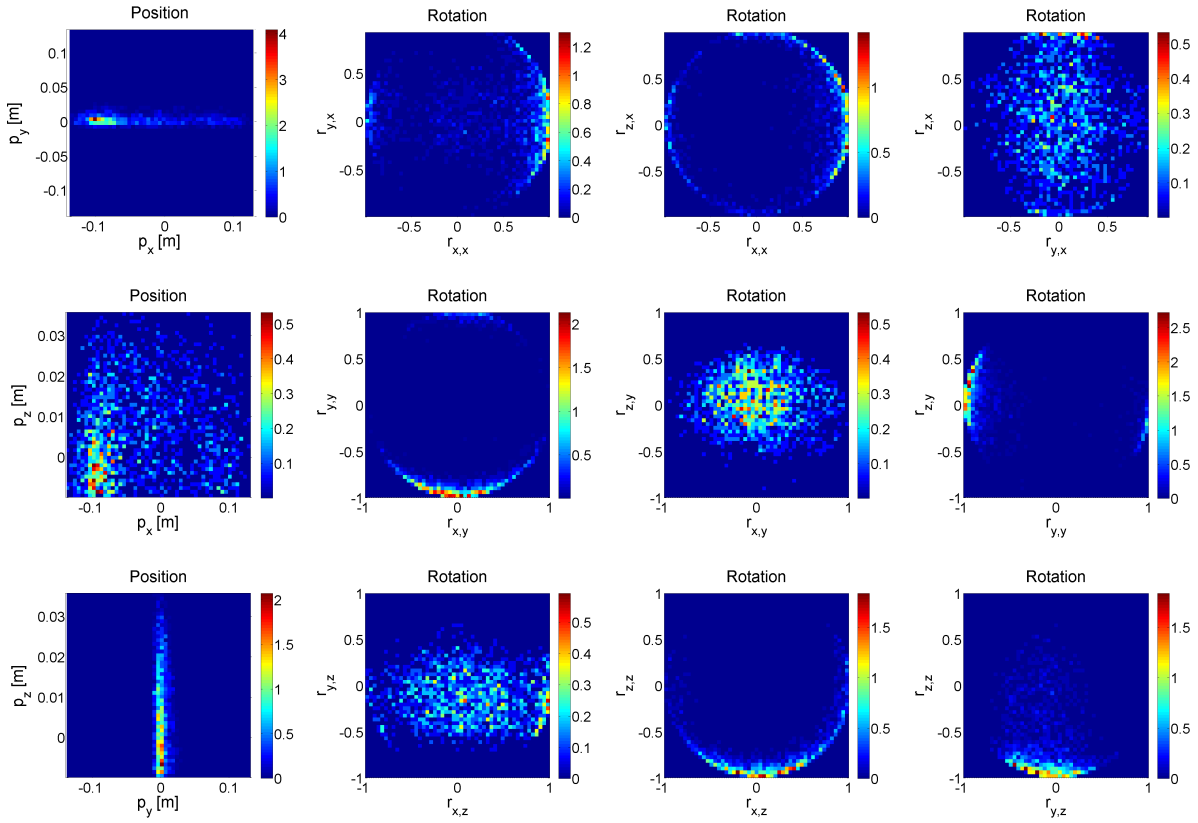


Figure 2.13: Histograms depicting the distribution the successfully simulated grasps in terms of 6D pose (for a detailed description see [Tho12]).

The task of the learning mechanism is to learn the associations between action-precondition pairs and their effects. Our approach is to encode the learning problem in terms of the inputs and outputs of a set of classifiers: for a particular action the effect on a single fluent can be predicted by a single classifier, taking as input a state description (the action-precondition pair). States are described using a deictic representation [1] which reduces the size of the state descriptions and supports generalization across states. The full set of changes to a state as a result of an action can then be constructed by combining (by conjunction) the changes predicted by each classifier. By using voted perceptrons [9] combined with a k-DNF kernel [12], this learning mechanism is able to learn to predict when actions in STRIPS domains will be successful and what their effects will be. However, the resulting action model is implicit to the voted perceptron model and cannot directly be used by automated planning systems. We therefore develop an additional step to extract STRIPS rules from the classifiers. First, we extract preconditions from the classifiers for individual fluents, using a technique similar to feature extraction methods such as SVM Recursive Feature Elimination (SVM-RFE) [10]. This results in a set of preconditions, where each precondition predicts change to a single fluent. Then, we use a heuristic method to combine the fluents into a set of effects, and the preconditions into a single precondition, giving a STRIPS rule for each action. In experiments with International Planning Competition (http://ipc.icaps-conference.org/) planning domains with noisy and incomplete observations, the resulting STRIPS action models are close to the actual domain models, and therefore the approach is highly suitable for use in planning applications.

Chapter 3

Conclusion

3.1 Links to other Workpackages

Deliverable D2.3.1 is based on input from WP2.1 and WP2.2 and provides input to WP3 for higher level structural bootstrapping tasks. The basic modules described here are the basis for the demonstration in WP5.2 and WP5.3.

References

[1] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI 1987)*, pages 268–272, 1987.

[2] 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*, 30(10):1229–1249, 2011.

[3] Alan D. Baddeley. *Essentials of Human Memory.* Psychology Press, Taylor and Francis, 1999.

[4] Andy Clark and Annette Karmiloff-Smith. The cognizer's innards: A psychological and philosophical perspective on the development of thought. *Mind & Language*, 8(4):487–519, 1993.

[5] Fernando J. Corbacho and Michael A. Arbib. Schema-based learning: Towards a theory of organization for biologically-inspired autonomous agents. In *Agents*, pages 520–521, 1997.

[6] Renaud Detry, Dirk Kraft, Oliver Kroemer, Leon Bodenhagen, Jan Peters, Norbert Krüger, and Justus Piater. Learning grasp affordance densities. *Paladyn Journal of Behavioral Robotics*, 2:1–17, 2011.

[7] W.D. Ellis, editor. *Gestalt Theory, A source book for Gestalt Psychology.* 1938.

[8] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[9] Yoav Freund and Robert Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–96, 1999.

[10] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[11] Andreas Jordt and Reinhard Koch. Fast tracking of deformable objects in depth and colour video. In Stephen McKenna, Jesse Hoey, and Manuel Trucco, editors, *Proceedings of the British Machine Vision Conference, BMVC 2011.* British Machine Vision Association, 2011.

[12] Roni Khardon and Rocco A. Servedio. Maximum margin algorithms with Boolean kernels. *Journal of Machine Learning Research*, 6:1405–1429, 2005.

[13] Oliver Kroemer, Renaud Detry, Justus Piater, and Jan Peters. Combining Active Learning and Reactive Control for Robot Grasping. *Robotics and Autonomous Systems*, 58(9):1105–1116, 9 2010.

[14] J. Y. Lettvin, H. R. Maturana, W. S. McCulloch, and W. H. Pitts. What the frog's eye tells the frog's brain. *Proceedings of the Institute of Radio Engineers*, 47:1950 – 1961, 1959.

[15] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 2(60):91–110, 2004.

[16] Cyberbotics Ltd. Webots reference manual, 2008. www.cyberbotics.com.

[17] Wail Mustafa, Mila Popović, Jeppe Barsøe Jessen, Dirk Kraft, Søren Maagaard Olesen, Anders Glent Buch, and Norbert Krüger. Using surfaces and surface relations in an early cognitive vision system. to be submitted.

[18] S.M. Olesen and S. Lyder. Real time 3D texlet extraction using GPUs and CUDA. *Master thesis at the Cognitive Vision Lab at the University of Southern Denmark, covil.sdu.dk/publications/ReportSimonSoerenRed.pdf*, 2011.

[19] J. Piaget. *The psychology of intelligence*. 1976.

[20] Mila Popović, Gert Kootstra, Jimmy Alison Jørgensen, Kamil Kuklinski, Konstantsin Miatliuk, Danica Kragic, and Norbert Krüger. Enabling grasping of unknown objects through a synergistic use of edge and surface information. *submitted*, 2011.

[21] N. Pugeault, F. Wörgötter, and N. Krüger. Visual primitives: Local, condensed, and semantically rich visual descriptors and their applications in robotics. *International Journal of Humanoid Robotics (Special Issue on Cognitive Humanoid Vision)*, 7(3):379–405, 2010.

[22] Sandor Szedmak, Hanchen Xiong, and Justus Piater. Learning shapes as directed closed surfaces. Technical report, IIS, University of Innsbruck, 2011. Draft of paper to be submitted. Attachment to Deliverable D2.1.1.

Attached Articles

[ADTW11] E.E. Aksoy, B. Dellen, M. Tamosiunaite, and F. Worgotter. Execution of a dual-object (pushing) action with semantic event chains. *2011 IEEE-RAS International Conference on Humanoid Robots*, 2011.

[BDPK11] Leon Bodenhagen, Renaud Detry, Justus Piater, and Norbert Krüger. What a successful grasp tells about the success chances of grasps in its vicinity. In *First Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB)*, 2011.

[BFW+12] L. Bodenhagen, A.R. Fugl, M. Willatzen, H.G. Petersen, and N. Krüger. Learning peg-in-hole actions with flexible objects. In *4th International Conference on Agents and Artificial Intelligence (ICAART) – Special Session on Intelligent Robotics (SSIR 2012)*, 2012.

[GKKed] F. Guerin, N. Krüger, and D. Kraft. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *IEEE TAMD*, submitted.

[JKP+12] J. Jorgensen, D. Kraft, J. Piater, H.G. Petersen, and N. Krüger. The complexity and potential of dexterous grasping examplified. Technical report, MMMI, University of Southern Denmark, 2012. Draft of paper to be submitted.

[KGP+11] 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.

[KPB+11] N. Kruger, M. Popovic, L. Bodenhagen, D. Kraft, and F. Guerin. Grasp learning by means of developing sensorimotor schemas and generic world knowledge. *Published at Convention AISB 2011 ('Artificial Intelligence and Simulation of Behaviour')*, 2011.

[MZPS] K. Mourao, L. Zettlemoyer, R.P. A. Petrick, and M. Steedman. Learning strips operators from noisy and incomplete observations. *submitted to AIII.*

[SKJP12] S. Szedmak, N. Krüger, J. Jorgensen, and J. Piater. Kernel methods for learning graspability function. 2012. Draft of paper to be submitted.

[Tho12] M.T. Thomsen. Introduction to feature to grasp association. *Internal Report*, 2012.

[USSM12] A. Ude, D. Schiebener, H. Sugimoto, and J. Morimoto. Integrating visual processing and manipulation for autonomous learning of object representations. In *Proc. IEEE Int. Conf. Robotics and Automation, Saint Paul, MN*, 2012.

This Paper acknowledges the EU-Projects IntellAct and Xperience.
The work on Semantic Event Chains (SECs) pursued here is part of IntellAct.
The work on executing pushing (using SECs) is part of Xperience

Execution of a Dual-Object (Pushing) Action with Semantic Event Chains

Eren Erdal Aksoy[1], Babette Dellen[1,2], Minija Tamosiunaite[1] and Florentin Wörgötter[1]
[1]Bernstein Center for Computational Neuroscience
University of Göttingen
Friedrich-Hund Platz 1, D-37077
Email: [eaksoye,minija,worgott]@physik3.gwdg.de
[2]Institut de Robòtica i Informàtica Industrial (CSIC-UPC),
Llorens i Artigas 4-6, 08028 Barcelona, Spain
Email: bdellen@iri.upc.edu

***Abstract*—Execution of a manipulation after learning from demonstration many times requires intricate planning and control systems or some form of manual guidance for a robot. Here we present a framework for manipulation execution based on the so called "Semantic Event Chain" which is an abstract description of relations between the objects in the scene. It captures the change of those relations during a manipulation and thereby provides the decisive *temporal anchor points* by which a manipulation is critically defined. Using semantic event chains a model of a manipulation can be learned. We will show that it is possible to add the required control parameters (*the spatial anchor points*) to this model, which can then be executed by a robot in a fully autonomous way. The process of learning and execution of semantic event chains is explained using a box pushing example.**

I. Introduction

If one wants to build robots which participate in everyday human life, learning from demonstration is perhaps the most practical paradigm. Although learning from demonstration has much advanced in recent years [1], [2], manipulation learning from demonstration has not yet come to its conclusion as here one has to bring together a sequence of demonstrated movements and task knowledge [3]. In our previous works [4], [5] we have introduced the so-called "Semantic Event Chain" (SEC) which is a compact and generic encoding scheme for manipulations. We have shown that the SECs can be used to allow an agent by observation to classify different manipulations and to categorize the manipulated objects based on their roles exhibited in the manipulation. Furthermore, we have demonstrated that an agent can learn an archetypical SEC model in an unsupervised way by watching about 10 demonstrations. The main advantage of this framework is that SECs link the signal domain (observed image sequences) to a symbolic rule-like domain encoding a manipulation in a highly invariant way, where – for a given manipulation – objects, poses, perspectives and trajectories can be interchanged to a very large degree. Thus, SECs provide one possible, quite efficient way to perform manipulation recognition and to learn a manipulation model.

SECs are essentially a symbolic representation encoding the manipulation by a temporal sequence of rules. Manipulations appear now in an abstract form, stripped from all pose and trajectory information and this makes it initially impossible "to invert the process" using a SEC for executing a manipulation. Thus, in the current paper we address the question how to actually do this and perform a manipulation starting from a SEC. Clearly, this requires that in the process of learning a manipulation model (learning the SEC) additional information must be stored, for example the start and endpoint of movement trajectories. But, because the SEC provides a temporal sequence of rules, we have well defined temporal anchor points when we have to store the additionally required trajectory information. Furthermore, as will be shown below, SECs also provide us with exact instructions, which spatial coordinates (spatial anchor points) are relevant for defining a movement.

The goal is to arrive at a fairly generic instruction set which allows "bringing two objects in close contact". Hence, to perform a basic dual-object action. Pushing two objects against each other but also pick one object up and placing it in contact to a second object fall into this category. To do this we will provide at the end a macro where we define an instruction set (D1-D8) which produces the basic movement segments for the execution. It is important to note that this macro will be applicable for all dual-object actions using the semantic event chains to structure and define the different execution primitives.

The structure of the paper is as follows. In Section II we discuss related works. In Section III, we briefly summarize our own prior work on how to learn a SEC from observation. Here we address the (new) problem, how to store information which is additionally required for execution. Next, in section IV, we describe how to actually execute a manipulation starting from a SEC showing simulation results. In Section V, the results are discussed and directions for future research are given. Finally, the work will be concluded in Section VI.

II. Related Work

In our previous works [4], [5] we have shown that SECs can both, classify manipulations and categorize manipulated objects in a model free way, needing prior representations neither for objects nor for actions. In [6] the authors built

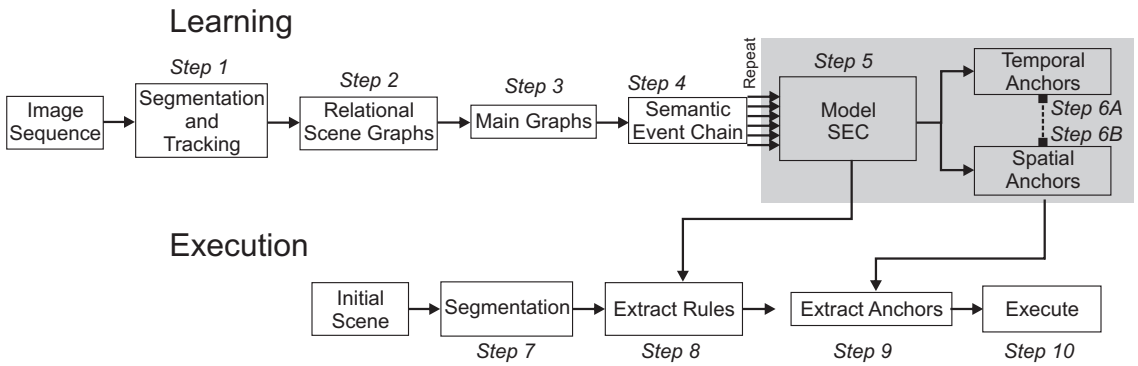


Fig. 1. Block diagram of the whole framework.

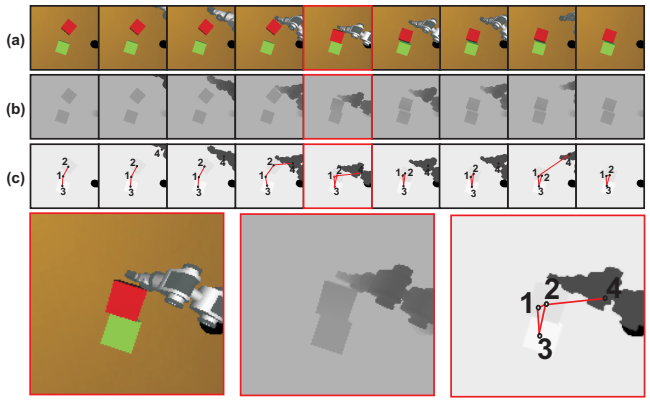


Fig. 2. Pushing action. **(a)** Original images from a movie recorded during the action. **(b)** Corresponding depth map from a range finder. **(c)** Corresponding HSV color based segmented images with extracted 3D scene graphs (See steps (1-2) in Fig. 1). Note that each object is represented by a unique segment label (e.g. 1, 2, 3, and 4 that represent *table*, *red box*, *green box*, and *robot arm*, respectively). Graph nodes represent the segments' centers and graph edges encode whether or not two segments touch each other in 3D. In red are indicated *Touching* relations between segments. The bottom part of the figure shows a magnified view of frames 5 from above.

a kernel based vectorial representation of event chains, which makes SECs more compatible with machine learning techniques. However, they have not addressed object categorization, learning and execution issues at all. Different from that, in the current work we will focus on execution of a learned pushing action by using SECs.

In the literature many works focus more on the (mechanical) aspects of controllability and planning of stable pushing actions [7], [8]. Such aspects are not in the core of our paper.

In [9] the authors showed how an agent can learn simple pushing actions on a toy object and then execute them as goal-directed behaviors. During the training phase, time evolution of the initial hand position and the direction of object displacement at the moment of contact were continuously recorded. As will be shown below, this is to some degree similar to our approach. In each trial the robot learns to map from initial hand position to the direction of object movement. However, the robot had only four possible initial positions which restricts the flexibility of manipulations in the execution phase of the learned maps. The high number of required trials (approximately 70) is another unrealistic drawback of this work.

In a different study [10] the problem of learning a general pushing rule has been addressed. The rule represents the relationship between the point and angle of push on the object's boundary and the observed object motion right after the pushing action. In the learning case the robot experimented with different pushing actions on different objects at different positions. The normalized retinal images of the experimental data served as input to a neural network to predict the object velocity in all directions. However, the input images had to be down-sampled to 20x15 pixels which causes much information loss. Moreover, in the testing case the robot has to drive an optimization process, the computational complexity of which is relatively high.

In [11] the authors described an on-line learning method for pushing an object to a desired (image) position. The system used past pushing operations to estimate future pushing actions. The main handicap of their approach is that the object is connected to the robot with a rotational point contact.

III. METHODS

In the current study, we perform execution of a pushing action by means of SECs. For this purpose we used the Webots software that simulates a 6 DOF Neuronics Katana robot arm. The experiment consists of two phases: Learning and execution (Fig. 1). In the first phase we perform manipulation demonstration. For this we already use the robot simulation and program it by hard-coding to push a red box to a green box on a table until they touch each other and then the robot is going to a home position. We could have used human demonstration instead, like in our older papers, but this is not of any relevance. The only thing needed is that demonstration is repeated using different setups.

Note, sophisticated object recognition is not part of this work. This is a difficult additional problem for which there are no generic solutions. In limited scenarios, such as those commonly used in state of the art robotics experiments, one could resort to conventional model-based object recognition methods. This step is simplified in our study and we get object

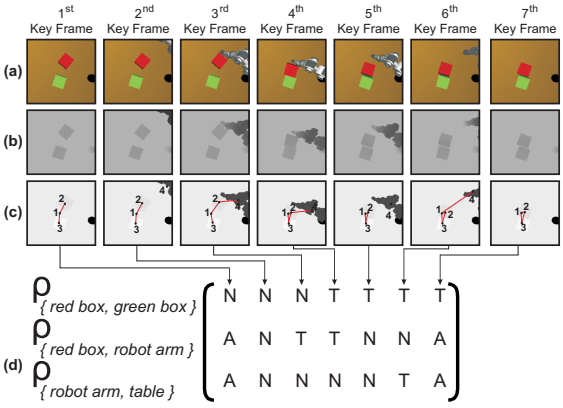


Fig. 3. Semantic event chain representation. **(a)** Original *"Key Frames"*. **(b)** Corresponding depth maps. **(c)** Corresponding HSV color based segmented images with extracted main graphs (See step 3 in Fig. 1). **(d)** Corresponding semantic event chain (See step 4 in Fig. 1), which is a sequence-table, where each entry encodes the spatial relations between each segment pair $\rho_{i,j}$ at each main graph. T means that segments touch (denoted by red edges), N means that there is no edge between two segments, and absence of a previously existing segment yields A.

identities (boxes, robot arm) by a unique color code, as more complex object recognition does not add to the relevant aspects of this study.

We also do not require that the red and green box should touch each other in some exact configuration. As a consequence, relative pose information can be neglected. However, it is also possible to store different touching types (i.e. pose information) between boxes in addition at the exact touching moment which is provided by the SEC. But, for the purpose of demonstrating the process of macronizing execution from a SEC it is sufficient to focus on the "ballistic push".

From these demonstrations a SEC-model is then learned. During learning also additional decisive information, for example relative coordinate frames and information about motion start and endpoints, is recorded. In the second phase (execution), we use the SEC and the additional information to let the robot execute a similar pushing action regardless of the initial state of the table.

A. Segmentation and SEC-generation (Steps 1-4)

Fig. 2 and Fig. 3 show a processing example of a manipulation resulting in its semantic event chain representation. We first extract all frames from the manipulation movie (Fig. 2 (a)) with corresponding depth maps from a range finder (Fig. 2 (b)). Frames are then segmented by a simple HSV color based segmentation algorithm, which allows for consistent marker-less tracking of each object (Fig. 2 (c)). Note that each object is represented by a unique segment label (e.g. 1, 2, 3, and 4). Once segments are calculated we drive a simple color based object recognition algorithm to replace those unique labels with object names (e.g. $table$, $red\ box$, $green\ box$, and tip of $robot\ arm$ instead of 1, 2, 3, and 4, respectively). After this step the agent knows which segment corresponds to which object. The scene is then represented by undirected and un-weighted graphs (Fig. 2 (c)). Nodes represent object center points and edges between nodes exist whenever two objects touch each other in 3D. Note, during a manipulation, graphs can change by continuous distortions (lengthening or shortening of edges) or, more importantly, through discontinuous changes (nodes or edges can appear or disappear). This happens when objects touch (or un-touch) each other. Such a discontinuous change represents, thus, a natural breaking point: All graphs before are topologically identical and so are those after the breaking point. Hence, we can apply an exact graph-matching method [12] at each breaking point and extract the corresponding topological main graphs. The sequence of these main graphs represents all structural changes (manipulation primitives) in the scene. The movie frames that hold such changes are called *"Key Frames"*. Fig. 3 (a-c) shows the *"Key Frames"* with corresponding depth map, segments, and main graphs for the action in Fig. 2. This type of graph representation is then encoded by the semantic event chain (Fig. 3 (d)), which is a sequence-table. Hence continuous time is now replaced by time-chunks where the same main graph persists until in the next chunk a new one appears. Each row in a SEC represents the temporally changing relations between one pair of objects in the scene, for example the first row in (Fig. 3 (d)) shows the relation between the red box and green box. There are three possible spatial relations defined between segments: *absence* (A), *no connection* (N), and *touching* (T). N means that there is no edge between two segments, corresponding to two spatially separated segments, and T represents segments that touch each other[1]. A special case exists when a segment has disappeared, which will be denoted by A.

Consequently, the complete image sequence, which has here roughly 320 frames, is represented by an event chain with a size of only 3×7. Note that several spatial relations, for example between *green box and robot arm* ($\rho_{3,4}$), are not included in this SEC since they do not contain any N-T or T-N transitions, which are decisive for a manipulation. Thus, we can always ignore such rows in the semantic event chain since they do not describe any manipulation relevant event.

These aspects are represented by steps (1-4) in Fig. 1, showing the block diagram of the complete algorithm.

B. Defining Temporal Anchor Points by Learning the SEC Model (Steps 5,6)

For learning we record the same pushing action from 10 different perspectives by changing the camera positions and extract the corresponding SECs. Fig. 4 shows the same, specific moment of the manipulation for each of the 10 different manipulation instances which allows us to get an impression of the level of perspective difference.

All movies used in this study can be found at http://www.dpi.physik.uni-goettingen.de/~eaksoye/Movies/PushingAction/. As described elsewhere [4], [5], we then calculate the pairwise percent-similarity between the

[1]In our older papers [4], [5] we had also used an *overlapping* relation (O), which in general, however, is not needed and is omitted here.

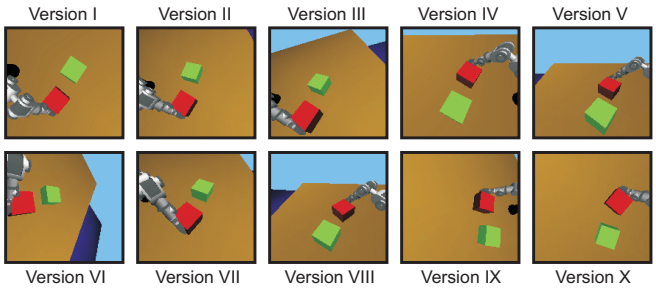


Fig. 4. Differences between the same moment of the pushing manipulation in all 10 different versions.

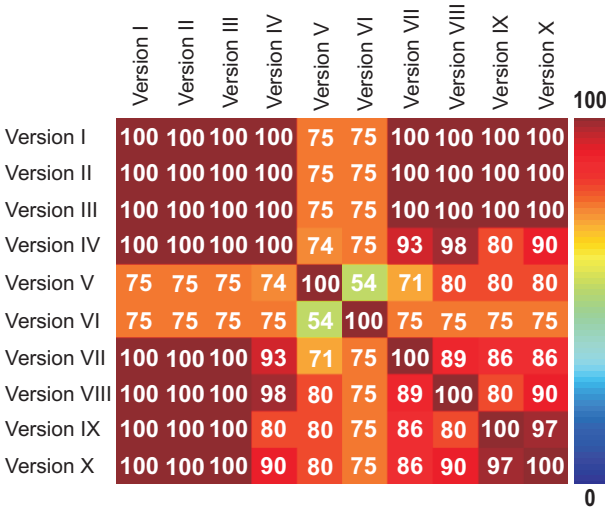


Fig. 5. Similarity values between 10 different versions of the pushing action.

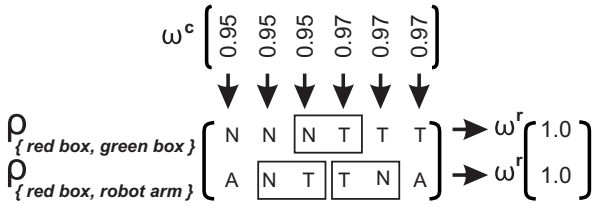


Fig. 6. The learned SEC model for the pushing action with corresponding normalized row (ω_i^r) and column (ω_i^c) weight values. Boxes indicate important transitions during this manipulation.

manipulations. Normally this step is used to classify different manipulations; here we use it to show that indeed a high mutual similarity exists between those 10 repetitions (see Fig. 5 for the confusion matrix), where one outlier with only 54% similarity occurred due to some error in the image segmentation.

Having assured that the individual SECs represent indeed the same manipulation we are allowed to perform a weighted average and extract all re-occurring rows and columns in the ten SECs. The resulting SEC is shown in Fig. 6. Weights ω represent the normalized occurrence frequency of a given row or column and are sometimes less than 1.0 due to the situation that not all rows and columns are present all the time in the individual SECs. This is also visible by comparing model (Fig. 6) with the single SEC in Fig. 3. Thus, the model represents the archetype-SEC for this particular pushing action.

Details of all those steps (Fig. 1) can be found in our previous works [4], [5]. At this stage it is important to note that the start points of each temporal chunk, given by the time moments of the different columns in the model-SEC, represent *temporal anchor points* (Key Frames of the movie sequence). The SECs, thus, solve a difficult chunking problem in a natural way: By these anchor points the different motor primitives needed for a manipulation are defined. Furthermore, these moments are also decisive for defining the spatial anchor points at the objects, needed to define and actually execute an action.

C. Defining Spatial Anchor Points

The temporal anchors tell us "what happened when?", but they do not yet answer the question "how did it happen?". In the most general case, we would also have to know, (1) which objects are moved, (2) how the final spatial configuration (relative poses) of the objects looks like, and how the movement trajectories are shaped requiring (3) start- and endpoints and (4) trajectory shapes.

We will in the following section show that an analysis at the temporal anchor points, hence of the Key Movie Frames, suffices to extract components 1-3: 1) objects involved, 2) required poses, and 3) movement start and end-points, which define motion segments. Only when wanting information about (4) the complete movement trajectory we need to analyze also movie frames *between* the key frames.

To keep the algorithm general we assume that only one prime mover exists (usually the robot arm), bimanual manipulations need a different treatment. As the robot arm can only do "one thing at a time" we can in general state that for all possible manipulations the manipulation is started when the robot arm produces the first N-T (non-touching to touching) transition in the SEC and that it ends when the arm produces a T-N transition. Thus, we first have to find the prime mover by analyzing the image segments. Furthermore, it is evident that all other existing N-T transitions are decisive for the manipulation. Hence, we need to analyze those – one after the other – too. Somewhat more unusual, we will also make use of the fact that *continuous contact* of prime mover with another object effectively makes the other object a secondary mover allowing us to use vector addition in task space for defining the complete motion path. For example, as long as the robot arm remains in contact with the red object (unchanging T relation), we can neglect the robot arm and just consider the newly resulting spatial relations of red object with other

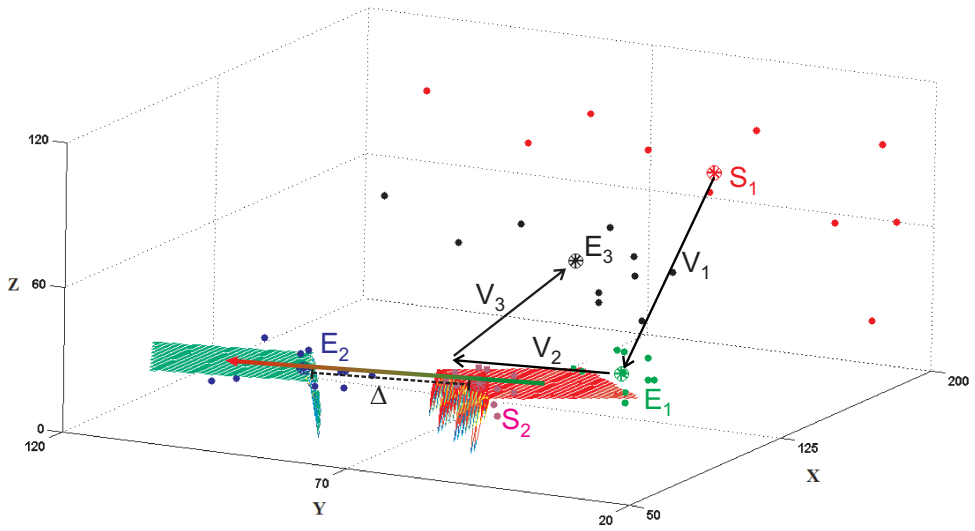


Fig. 7. Start (S_1, S_2) and end (E_1, E_2, E_3) point distribution as provided by the training dataset. Starred points are the average locations. Robot manipulator travels along the path given by vectors V_1, V_2, V_3. The coordinate origin is at the center of the red object. The motion vector for pushing is depicted by the color-changing vector that connects red with green object. Distance Δ is defines as $|E_2 - S_2|$ and vector V_2 has length Δ.

objects (here the green object) as spatial anchors[2]. Note, this reflects also the aspect of a robot using a tool. As long as the machine holds the tool, the robot's body is essentially extended and the tool defines now the end-effector of the machine [13].

Analyzing the Demonstration Examples: Let us first analyze the 10 demonstration pushes to see how the movement segments actually looked like. By D1-D8 we denote in the following those constraints which are used to actually define the movement segments for execution. As we do not need poses (ballistic push!), we did not implement any pose estimation steps.

To find the prime mover, we take the first N-T transition $(N_{2,2} - T_{2,3})$ in the model-SEC (Fig. 6). Here we use conventional row,column indices just for making it easier to find the entries in the SEC. Now we subtract the image segment configuration at key frame at $N_{2,2}$ from that at $T_{2,3}$, leading to a difference image only at the robot-arm image segment as the red box has not yet moved.

D1:*Thus, we obtain as prime mover the "robot arm segment".*

Then we consider the actual start points of the movement (red points in Fig. 7), which are widely distributed. The average is given by the starred red point.

D2: *Hence there are essentially no constraints on the starting point S_1 of the complete sequence.*

Next, we record at key frame at $T_{2,3}$ the coordinates of the red object at the touching point (see green points in Fig. 7).

D3: *This defines the endpoint E_1 for this specific motion segment V_1.*

We need to make sure that execution can cope with all kinds of different spatial configurations of robot arm and object. This requires defining a coordinate system which allows for such a generalization. To this end we use as the coordinate origin the segment center of the first touched object. This definition holds true for all conceivable basic single- or dual-object manipulation actions as *relations* between objects are decisive for the manipulation(s). Hence, we can always fix the origin on the first one touched and define coordinates relative to this[3], where we use any generic cartesian coordinate system just keeping it fixed for the remainder of the process.

D4: *Thus, the center of the first touched object defines the coordinate origin.*

The second found N-T transition concerns the red and the green object $(N_{1,3} - T_{1,4})$. As there is no change between the relation of robot arm and red object (relation remains $T_{2,4}$) we have indeed found a "secondary" mover (the red box) and a second touched object (the green box).

The segment center of the green object (the second touched object) defines together with the coordinate origin (center of red object) the so-called *dual object connection vector* (short: connection vector). Also this definition holds for all dual-object manipulations where a first object is supposed to make contact with a second object. In all these cases the first object must travel along a path (vector) that connects it to the second one. Clearly, in many dual-object manipulations additional difficult pose-constraints may arise, but the general connection vector will remain the same.

D5: *Thus, the connection vector is spanned between the*

[2]There might be some complicated manipulation actions where the arm (or hand) touches (or picks up) a second object before releasing the first. In this case, the argument about secondary mover would not hold. But such manipulations are uncommon and even for a human quite difficult. Hence, we do not consider them.

[3]Most basic, uni-manual manipulations are performed either at one object, leading to some configuration change at the object, or at two objects, where the first touched object is combined with the second one. Other manipulations, where more objects are directly involved are very rare (e.g. grasping two objects keeping both in the hand and combining them with a third one) or they can be considered as a chain of single- or dual-object manipulations.

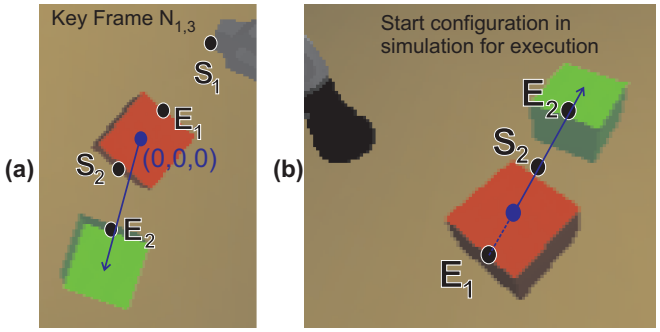


Fig. 8. Start and end points as given by demonstration (A), and as calculated for execution (B).

centers of the first and second touched object. Movement segment V_2 should follow the direction of this vector.

Now we need to define the path length. So far the definitions do not require any prior knowledge about the actual action to be performed. They hold for pushing as well as, for example, for pick-and-place actions. The fact that we want to perform a pushing action only comes in now: Similar to above, we record for key frame at $T_{1,4}$ the coordinates of the red *and* green objects at their touching point. They are shown back-projected onto the start frame in Fig. 7 (pink and blue). One can see that for a push, start and endpoints E_1, S_2, E_2 of the motion segments are roughly aligned with the connection vector (see Fig. 8 A and Fig. 7).

D6: *Points E_1, S_2, E_2 can be computed from the 3D-coordinates representing the intersection of the connection vector with the edges of the objects (Fig. 8 B).*

D7: *From this, we also note that the distance $\Delta = |E_2 - S_2|$ defines the length of the second motion segment V_2. Its direction is given by the connection vector.*

The core of the manipulation ends at the $T_{2,4} - N_{2,5}$ transition of robot arm with red object. The final homing motion of the robot arm, which follows thereafter, is not relevant for the manipulation and can be performed in any possible way. We look at the now following $N_{2,5} - A_{2,6}$ transition at the prime mover and plot the end points E_3 from $A_{2,6}$ (black points in Fig. 7) producing a set of actually observed final endpoints of the robot arm, which are also widely distributed.

D8:*Any endpoint for the motion can be used as long as the robot arm withdraws from the red object in a collision free way.*

D. Execution (Step 8)

The actual execution now is simple. For this, the model-SEC is used and every transition is treated like a rule. Constraints D1-D8 are attached to the transition rules as defined above.

For example the first N-T transition corresponds to a rule that demands that some movement by the prime mover should take place such that at the end the robot arm touches the red box and so on.

Thus, a new visual scene is presented to the system and segmented as usual. Robot arm, red box, and green box are recognized by their color or by any other object recognition algorithm. The model-SEC is split into its rules and the *actual* movement sequence is prepared by calculating the movement segments relative to the target objects. Start point S_1 is given by the momentary location of the robot arm. Again we use as coordinate origin the center of the red object and the connection vector points to the center of the green object. This origin- and vector-definition holds for all dual-object manipulations. For the specific purpose of pushing, and as explained above, the respective start and endpoints E_1, S_2, E_2 are now computed from the 3D-coordinates representing the cross-section of the connection vector with the edges of the objects (Fig. 8 B). The movement amplitude for the second N-T transition is in the same way given by $\Delta = |E_2 - S_2|$. We note that the touching point of the second object can be a bit over-estimated by this procedure if the diagonal of the object is aligned with the connection vector. In this case we will get a bit of a "push-second-object-away" when executing the action. This shows that pose estimation will at some point have to be added, too. Movement from S_1 to E_1 is defined by any collision free trajectory all other motion segments are straight. The last motion segment is a homing movement to any desired endpoint (E_3).

Without having to explain the details, execution now proceeds by following the N-T or T-N transitions from the model-SEC using conventional inverse kinematics for the Katana arm by vector addition of all motion segments until the sequence of motion segments has been consumed.

It is important to note the the robot has now immediately also a means to check whether the outcome of its actions are correct. After each movement of the arm, the machine needs to check the resulting relational changes between the image segments. These should match the changes in the model-SEC (see [5] for an example).

IV. Simulation Results

We let the agent realize the pushing action considering the learned model-SEC, and the motion segments as defined above. Fig. 9 (a-c) shows how the *robot arm* pushes the *red box* to the *green box* even if the object locations are different. In Fig. 9 (d-e) we used a red sphere and a bigger red box as pushable objects. In such cases the *robot arm* can still execute the manipulation. In Fig. 9 (f) a red cone and one more blue sphere were used. Finally, the robot arm chose and pushed the red conic to the bigger green box. All those simulation results, with corresponding depth, segment, and graph representations, can be found at http://www.dpi.physik.uni-goettingen.de/~eaksoye/Movies/PushingAction/. Finally we performed a self-check and Figure 10 shows the SEC obtained from the last execution example in Fig. 9, which is very similar to the model-SEC by which the robot can accept its own execution as correct. This is true for all executed examples.

These results show that with such a semantic representation the agent can learn and imitate a ballistic pushing manipulation independent of object shapes and positions.

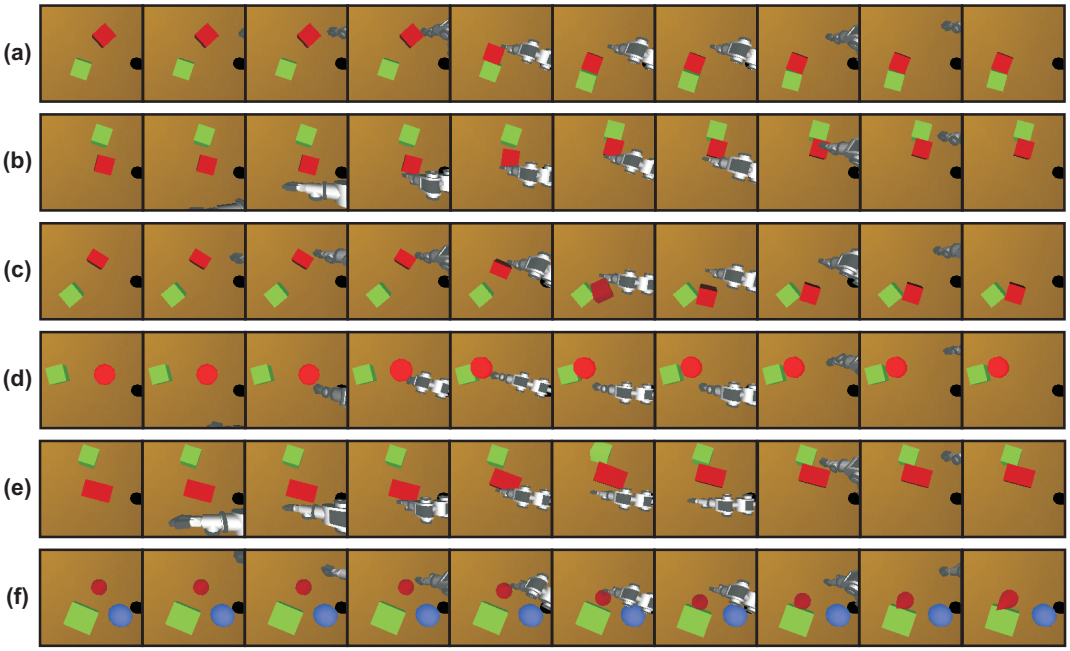


Fig. 9. Execution results. (a-c) *Robot arm* pushes the *red box* to the *green box* even if the object locations are different. (d-e) A red sphere and a bigger red box are used as pushable objects. (f) A red cone is used as a pushable object and one more blue sphere is added in the scene.

$$\begin{matrix}\rho_{\{\textit{red box, green box}\}}\\ \rho_{\{\textit{red box, robot arm}\}}\\ \rho_{\{\textit{robot arm, table}\}}\end{matrix}\begin{pmatrix}\mathrm{N} & \mathrm{N} & \mathrm{N} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T}\\ \mathrm{A} & \mathrm{N} & \mathrm{T} & \mathrm{T} & \mathrm{N} & \mathrm{N} & \mathrm{A}\\ \mathrm{A} & \mathrm{N} & \mathrm{N} & \mathrm{N} & \mathrm{N} & \mathrm{T} & \mathrm{A}\end{pmatrix}$$

Fig. 10. SEC obtained from execution of the last example in Fig. 9.

V. Discussion

In this paper we have introduced a novel representation for the execution of manipulations by using the semantic event chains, which focuses on the spatial relations between objects in a scene. The representation generates column vectors in a matrix where every transition between neighboring vectors can be interpreted as an action rule, which defines which object relations have changed in the scene. In the first step, the approach learns from demonstration an archetypal event chain (model-SEC) consisting only of consistently repeated rows (spatial relations) and columns. Apart from the demonstration no other supervision is needed in this step, hence SECs are learned in a model-free way. In the second step, the learned rules are enriched by determining the movement segments by which the manipulation can be executed regardless of the configuration of the objects in a scene. Execution can then follow the enriched SEC rules and the robot can test its own success by checking the SEC, which results from execution, against the model-SEC.

To our knowledge this is the first approach which uses a learnt abstract symbolic representation for manipulation learning, execution, and self-recognition. We are aware of the fact that the algorithm uses some simplifications such as no object dynamics and pose estimations. Due to this fact, in some cases it was observed that the object could not be pushed in the desired direction, because of wrong object and/or gripper poses and the frictional restrictions both on the background and object surface.

It is important to note that this procedure can be macronized for all *dual-object manipulations*. In a very abbreviated form the instructions for such a macro would read:

1) `Identify prime mover.`
2) `Identify first touched object by first N-T transition and set coordinate origin.`
3) `Define first motion segment`
4) `(Extract relative poses between prime mover and first object, if needed).`
5) `Identify second touched object and fix connection vector and coordinate system.`
6) `Define second motion segment for second N-T transition relative to this coordinate system. (For pushing do this by cross-sectioning with object borders).`
7) `(Extract relative poses between objects involved, if needed).`
8) `Define third motion segment (home).`

Such a macro can be enriched by adding pose information from a pose estimation algorithm where required. This would be needed for a pick&place manipulation (which is also a dual object manipulation), where the final resulting relative pose of the two combined objects is most of the time important. Aspects of grasping an object (e.g. grasp preparation and the performing of a grasp) are not considered at all in this

Fig. 11. SEC for the action "cutting a carrot with a knife".

framework. Grasping is a very difficult technical problem but for manipulation actions it takes usually just a preparatory role. We do not wish to downgrade the importance of this role but the actual outcome of the manipulation is in most cases only in a secondary way affected by the way an object is grasped. Clearly, if the grasp is totally unsuitable a pick&place action will fail. But these considerations must happen before the first N-T transition in the SEC and are not part of this paper.

Note, the presented approach can also be extended to more general manipulation tasks, e.g. "cutting a carrot with a knife". In such a SEC we would observe more relational changes compared to the pushing example as the newly cut carrot pieces have to be also included into the representation. The SEC for cutting two pieces off the carrot is given in Fig. 11.

For execution purposes we again have to analyze N-T and T-N transitions, which in given cutting scenario will emerge only between *hand and knife*, *knife and carrot*, and *knife and the two new appearing pieces*. Therefore, fourth, sixth, and seventh rows of the SEC given in Fig. 11 can be ignored for clarity. In addition to the discussed pushing example, for the cutting scenario grasping of a knife will be required, where the SEC will provide the temporal anchor point for extracting relative pose between *hand and knife* at the hand-knife N-T transition. The relative pose between *knife and carrot* will be important at the transitions N-T in the relation knife-carrot. Also movement trajectory information for the cutting (actually, sawing) movement should be attached to the SEC between the N-T and T-N transitions in the relation knife-carrot, which will then be used in execution. This shows that the same anchoring process can also be performed for other actions. Clearly, this involves very difficult technical steps of pose estimation and trajectory shaping.

VI. Conclusion

In the current study we have shown how a learnt SEC can be used to "push one object against another one." We tried to make clear that this example generalizes to all dual-object actions, where for different instantiations different pieces of information (e.g. addition pose information, etc.) have to be stored. The anchor points for this information, however, exist as demonstrated in the current study and the inversion of an event chain is possible. Work has started to address this problem for a large ontology of manipulation actions [14].

Acknowledgments

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 and grant agreement no. 269959, IntellAct. B.D. acknowledges support from the Spanish Ministry for Science and Innovation via a Ramon y Cajal Fellowship.

References

[1] 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.

[2] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *Int. J. Hum. Robot.*, vol. 5, pp. 183–202, 2008.

[3] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zollner, "Incremental learning of tasks from user demonstrations, past experiences, and vocal comments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 2, pp. 322–332, 2007.

[4] 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, ICRA2010 Alaska, USA*, 2010.

[5] 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 (IJRR), Special Issue on 'Semantic Perception for Robots in Indoor Environments' (In press)*, 2011.

[6] L. Guoliang, N. Bergström, C. H. Ek, and D. Kragic, "Representing actions with kernels," in *IEEE/RSJ International Conference on Intelligent Robots ans Systems (IROS)*, 2011, (To appear).

[7] K. Lynch and M. Mason, "Stable pushing: Mechanics, controllability, and planning," in *Algorithmic Foundations of Robotics*. Boston, MA: A. K. Peters, 1995, pp. 239–262.

[8] Q. Li and S. Payandeh, "Manipulation of convex objects via two-agent point-contact push," *Int. J. Rob. Res.*, vol. 26, pp. 377–403, April 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1274664.1274673

[9] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, G. Sandini, and G. S, "Learning about objects through action - initial steps towards artificial cognition," in *In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA*, 2003, pp. 3140–3145.

[10] D. Omrcen, C. B. T. Asfour, A. Ude, and R. Dillmann, "Autonomous acquisition of pushing actions to support object grasping with a humanoid robot," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Paris, France, 2009.

[11] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini, "A vision-based learning method for pushing manipulation," in *In AAAI Fall Symposium Series: Machine Learning in Vision: What Why and*, 1993.

[12] M. F. Sumsi, "Theory and algorithms on the median graph. application to graph-based classification and clustering," Ph.D. dissertation, Universitat Autonoma de Barcelona, 2008.

[13] 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*, vol. 57(4), pp. 420–432, 2009.

[14] F. Wörgötter, E. E. Aksoy, N. Krüger, J. Piater, A. Ude, and M. Tamosiunaite, *Robotics and Autonomous Systems, (submitted).*

What a successful grasp tells about the success chances of grasps in its vicinity

Leon Bodenhagen, Renaud Detry, Justus Piater and Norbert Krüger

***Abstract*—Infants gradually improve their grasping competences, both in terms of motor abilities as well as in terms of the internal shape grasp representations. Grasp densities [3] provide a statistical model of such an internal learning process. In the concept of grasp densities, kernel density estimation is used based on a six-dimensional kernel representing grasps with given position and orientation. For this so far an isotropic kernel has been used which exact shape have only been weakly justified. Instead in this paper, we use an anisotropic kernel that is statistically based on measured conditional probabilities representing grasp success in the neighborhood of a successful grasp. The anisotropy has been determined utilizing a simulation environment that allowed for evaluation of large scale experiments. The anisotropic kernel has been fitted to the conditional probabilities obtained from the experiments.**

We then show that convergence is an important problem associated with the grasp density approach and we propose a measure for the convergence of the densities. In this context, we show that the use of the statistically grounded anisotropic kernels leads to a significantly faster convergence of grasp densities.

I. Introduction

When using already made grasping experience with a specific object there is no way to repeat the exactly same grasp due to uncertainties on pose estimation as well as the actual grasping process as such. Hence assumptions about grasps likely to be successful in the vicinity of grasps already tested and memorized are required even when exactly the same grasp is repeated for the same object in a new situation.

To tackle this, recently the concept of grasp densities [3] has been introduced which describes grasp affordances associated with specific objects in a probabilistic way. It is based on a kernel density estimation [9] in which the success likelihood of already tried grasps is described by 6-dimensional kernels (illustrated in figure 1 bottom left). The set of all grasp affordances associated with an object can then be expressed as the sum of these spatially extended 6D kernels (see figure 1 top right). Grasp densities have proven very useful in applications since they allow for the formulation of optimal grasps under constraints (see figure 2). In addition, since grasp densities reflect success likelihoods of each grasp, these can be used in higher level processes such as action sequence planning.

L. Bodenhagen and N. Krüger are at University of Southern Denmark. Email: lebo@mmmi.sdu.dk
R. Detry is at the Royal Institute of Technology, Sweden
J. Piater is at the University of Innsbruck, Austria

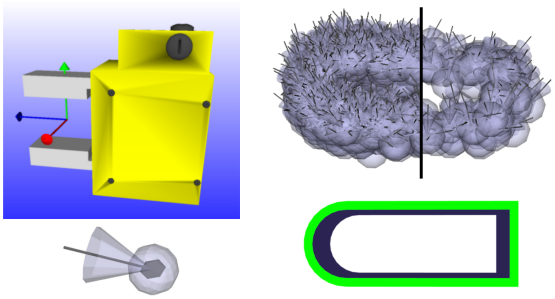


Fig. 1: Illustration of the parallel-jaw gripper (top left) and the 6D-kernel associated with the grasp (bottom left). And a visualization of a grasp density associated with an artificial object (top right). The left part shows all kernels, the right part only 10% of the kernels in order to provide a better overview. The original object is shown at the bottom (right).

However, currently two severe problems exist when using grasp densities. First, the exact shape of the kernels being used in previous work is only weakly motivated and is in particular isotropic (see figure 1 bottom left). Nonetheless high structural dependencies can be assumed to exist in the space of grasps associated with objects due to the intrinsic regularities of objects. The understanding of these regularities and how they can be expressed in kernels is an interesting topic in itself. In this context the first contribution of our paper is to give a statistical justification of the shape of the success likelihood of grasps in the vicinity of already successfully tested grasps. This is done by means of statistically derived conditional probabilities in grasp simulations.

As we will show in this paper, a second problem of the grasp density approach [3] is a rather slow convergence of the algorithm when the complete set of affordances is supposed to be represented as it requires a large set of grasp attempts. In this paper we give evidence that by using the statistically derived anisotropic kernel, we can speed up the convergence of the algorithm significantly.

The paper is structured as following. In section II, we introduce basic notations and methods used in this paper. In section III, we introduce the results on our statistics which motivate the choice of a new anisotropic kernel. The adaption of the new anisotropic kernel to the statistical results is outlined in section III

Fig. 2: Grasp success likelihoods learned by means of exploration are represented in green. A local maximum indicates optimal grasp points. Constraints of graspability (e.g., on invoked by workspace constraints as indicated by the sharp green border) can be easily integrated.

as well as its application to grasp density estimation.

II. Methods

In the following we provide a detailed overview of the different methods used throughout this paper. In section II-A our parametrization of grasping actions is defined, the association from actions to objects using grasp densities is described in section II-B. Two different designs for anisotropic kernels are outlined in section II-C. The simulation environment wherein actions are performed is introduced in section II-D and in section II-E the choice of bandwidths for an isotropic kernel is discussed.

A. Grasps and their transformations

A grasping action, A, is in this context defined as a point in the special Euclidean space, $A \in SE(3)$ and defines the transformation from the object reference frame to the tool which performs the grasp. In this work, the tool is considered to be a parallel finger gripper — the setup is illustrated in Fig. 3.

Based on a set of evaluated and successful grasps, $\mathcal{S} = \{A_1, \ldots, A_n\}$, for one specific object it is investigated if a grasp still would be successful when it becomes transformed locally:

$$P(T_{RBM}(A) \text{ is successful} | A \text{ is successful})$$

where $T_{RBM}(x)$ denotes a rigid body motion applied to the action A, thus

$$T_{RBM} : SE(3) \rightarrow SE(3)$$

Note that A and T_{RBM}, although both elements of $SE(3)$ have two separate meanings, A representing a grasp and T_{RBM} a rigid transformation.

The success of a transformed action, $T_{RBM}(A)$, is estimated by comparing it with all grasps in $\mathcal{S}$ using the grasp density (introduced in section II-B). To be able to do a reasonable comparison it is required that the density covers the entire object. It is not feasible to evaluate every $T_{RBM}(A)$ physically or even in a simulator as this still would be far too time consuming.

B. Grasp Densities

A grasp density models the distribution of successful grasps relative to an object. A density is defined as a probability density function, $p_{X|O=s}(x)$, where $X \in SE(3)$ represents a gripper pose and $O \in \{s, f\}$ is the outcome of a grasp which can be either success or failure. The value of a grasp density at a concrete pose $x \in SE(3)$ is proportional to the likelihood of a successful grasp when the gripper is moved to this pose and closed. A typical applications for grasp densities is for example the search for a local maximum which provides the user with a grasp hypothesis with a high likelihood of being successful (see figure 2). External constraints, e.g. due limited workspace of a robot, can easily be integrated by limiting the search space (see figure 2).

A grasp density $d(x)$ is estimated using kernel density estimation (see also [2], [9]):

$$d(x) = \sum_{i=0}^{n} w_i \mathbf{K}_{\mu_i,\sigma}(x) \tag{1}$$

where w_i is a weight that compensated the impact of the sampling strategy. The kernel $\mathbf{K}$ is defined as a product of a trivariate Gaussian kernel, $\mathbf{N}$, for the position and a orientation kernel $\mathbf{\Theta}$ defined on $SO(3)$ by the von-Mises-Fisher distribution [5] (see also figure 1):

$$\mathbf{K}_{\mu,\sigma}(x) = \mathbf{N}_{\mu_t,\sigma_t}(\lambda)\mathbf{\Theta}_{\mu_r,\sigma_r}(\theta) \tag{2}$$

where

$$\mathbf{\Theta}_{\mu_r,\sigma_r}(\theta) = \frac{1}{2} C_4(\sigma_r) \left(e^{\sigma_r \cos(\beta)} + e^{-\sigma_r \cos(\beta)} \right) \tag{3}$$

$$\beta = \cos^{-1}\left(\mu_r^T \theta\right) \tag{4}$$

where σ_t denotes the width of the kernel for the position and σ_r denotes the width of the kernel for the orientation. Similar μ_t and μ_r denote the mean values in $SE(3)$. λ and θ are the pure positional respectively orientional part of $x = (\lambda, \theta)$ — orientations are in all cases represented using quaternions. $C_4(\sigma_r)$ is a constant which ensures that the density integrates to one. Note that $\mathbf{\Theta}_{\mu_r,\sigma_r}(\theta)$ basically depends on the angle between two quaternions which is a scalar (for further details, see [3]).

C. Anisotropic kernels

One property of the kernel $\mathbf{K}$ as introduced in equation 2 and in more detail in [3] is that both the position and the orientation are modeled isotropically (see figure 1a). Different strategies to model an anisotropic kernel, $\hat{\mathbf{K}}$, which expresses the structural properties of successful grasps in a neighborhood have

been considered. Note that an anisotropic kernel can be defined as a sum of multiple isotropic kernels with constant widths:

$$\hat{\mathbf{K}}^{S}_{\mu,\sigma}(x) = \sum_{j=0}^{m} w_j \mathbf{K}_{\mu_j,\sigma_j}(x) \tag{5}$$

allowing for an approximation $\hat{d}(x)$ of $d(x)$ by

$$\hat{d}(x) = \sum_{i=0}^{n} w_i \hat{\mathbf{K}}^{S}_{\mu,\sigma}(x) \tag{6}$$

Although the approach in equation (5) is computationally not optimal it can be used to investigate the impact of using anisotropic kernels for the generation of grasp densities and it does not imply any restriction on the shape of the anisotropic kernel.

The alternative is an analytic expression of $\hat{\mathbf{K}}_{\mu,\sigma}(x)$. Remembering that the value of original kernel used for the orientation is dependant on the angle between two quaternions (see equation 3). This angle can be weighted which leads to the following formulation of the kernel:

$$\begin{aligned}\hat{\mathbf{\Theta}}_{\mu_r,\sigma_r}(\theta) = \frac{1}{2}C_4(\sigma_r)\left(e^{\sigma_r \cos(w_{\mu_r}(\theta)\beta)}\right) + \\ \frac{1}{2}C_4(\sigma_r)\left(e^{-\sigma_r \cos(w_{\mu_r}(\theta)\beta)}\right)\end{aligned} \tag{7}$$

where the angle is weighted by $w_\mu(\theta)$, which is similar to Mahalanobis distance measure [8]:

$$w_{\mu_r}(\theta) = \sqrt{(\theta - \mathbf{0})\, S^{-1} R_{\mu_r}\, (\theta - \mathbf{0})'} \tag{8}$$

where S is a diagonal matrix that describes a 4D-ellipsoid and R_{μ_r} represents the rotation that aligns the ellipsoid with the mean-orientation of the kernel. Similarly the Gaussian kernel $\mathbf{N}$ can be defined using a covariance matrix to use the Mahalanobis distance rather than the Euclidean. In this paper we describe the basis for such an approximation as outlined in the discussion.

D. Simulator and Simulations

As a large set of evaluated grasps is required to achieve complete coverage of grasp affordances a simulator [7] has been used (in total we simulated about 10.000.000 grasps). The setup is illustrated in figure 3. The use of a simulator allows us to evaluate large sets of grasps efficiently while avoiding the overhead of using a real setup and circumventing the introduction of errors by usage of a pose estimation algorithm to obtain the pose of the object.

In the simulator both objects and grippers are defined by geometric models as well as mass and friction information. While the material of the gripper is known, we estimate the mass of the objects and assume them to be made of plastic. For the gripper it is in addition ensured that the realistic constraints on the position, velocity and acceleration of the fingers are maintained.

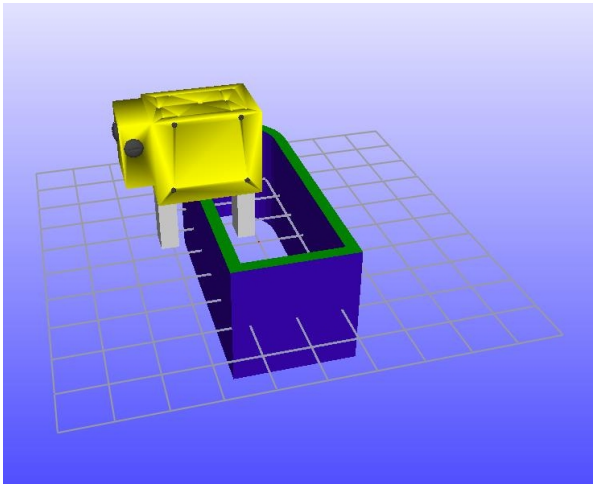


Fig. 3: Screenshot from the simulator.

The positions of the grasps that are to be simulated are obtained by defining a 6D grid covering all the poses that are in the vicinity of the object. The resolution of this grid needs to be limited in order to ensure that simulation still is tractable. In our simulation, we used a grid of 10mm and approximately 15 degrees.

The simulator models the interaction between the tool, a simulated Schunk PG70 parallel-jaw gripper with a maximum finger distance of 70mm, and the object during the grasping process using a physics engine. The execution of a grasp is finished when either the gripper is closed entirely or the object prevents it from doing so (see figure 3). Grasp hypotheses that would lead to collisions between gripper and object beforehand are discarded. Whether a grasp is successful or not is estimated by calculating the grasp force wrench and determining whether the grasp can counteract gravity (see also [4]). If that is the case, the grasp is considered to be successful. We only consider the force wrench as the simulation is based on hard contacts. Therefore a stable grasp may consist of only two contact points, although the object might rotate freely on the axis defined by the two contact points.

The four different objects used in our simulations are shown in figure 4. The screwdriver and the coffee-mug have been found on the web[1]. Their size matches the size of common real world objects. The screw driver is approx. 280mm long and up 30mm wide, the body of the cup is approx. 60mm wide and 80mm high. The elongated D-shaped object (top right in figure 4) and the cone are purely artificial objects. The width of the cone is 80mm at it's base and 20mm at the top while the overall length is 400mm. The dimensions of the elongated D are approx. $140\text{mm} \times 140\text{mm} \times 350\text{mm}$.

Figure 5 shows a visualization of a grasp density that has been created for the cone. For the projections of the density a plane has been defined that contains the main axis of the cone and is parallel to the image

[1] http://sketchup.google.com/3dwarehouse/

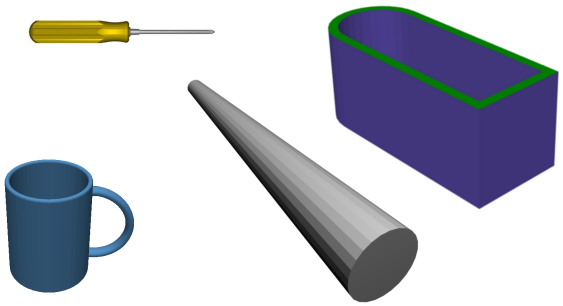
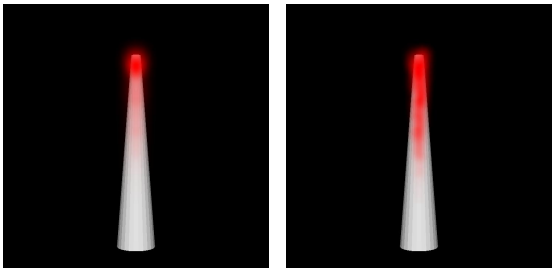


Fig. 4: The different objects used for the simulations. Some of the objects have been scaled for illustrative purposes.



(a) (b)

Fig. 5: An example of a grasp density. The more opaque the red color is, the higher the values of the density is at this point. (a) illustrates the distribution of all grasps, (b) illustrates the distribution of all grasps that are oriented vertical to the image plane and aligned with the main axis of the object.

plane of a virtual camera. Subsequently the density has been projected on the plane in red. The opaqueness of the color indicates the value of the density where completely opaque refers to the maximum and completely transparent regions indicate that no successful grasps have been experienced there. Note that the projections have been normalized individually to guarantee that the highest value of projection saturates the red colore channel. In figure 5a all grasps have been projected showing that most successful grasps occurred at the narrow end of the cone. Figure 5b show the distribution of all grasps that are oriented vertical to the image plane and aligned with the main axis of the object.

E. Optimal isotropic kernels

The grasps obtained using the simulator are used to create a grasp density. This process requires that a suitable kernel width is selected. There is a trade-off between having small kernels requiring a fine-grained density using a large number of kernels to reach a full coverage but allows to have relatively sharp borders separating successful and non successful grasps and having wide kernels which ensure that entire graspable part of the object is covered by the density using fewer samples with the cost that borders between successful

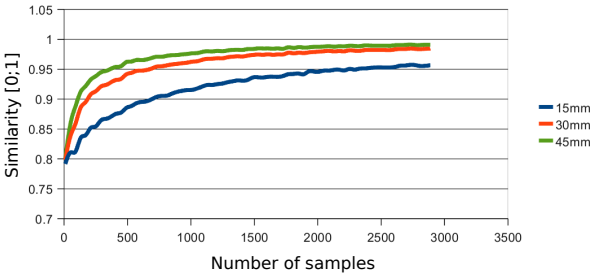


Fig. 6: Bootstrapping applied to a density using three different kernel widths.

and non-successful grasps are blurred. In this section we discuss this trade-off aiming at estimating a suitable kernel width.

Since the set of true grasp affordances is not directly accessible, it is difficult to select proper widths. Therefore the concept of bootstrapping (see [6]) is utilized in order to achieve a convergence measure s. Bootstrapping is in general a strategy to estimate statistical properties of any measurement by estiming these properties directly on samples of the distribution that approximates the measurement. The measurement in our case is the overall graspability of the object, estimated by a grasp density. Each grasp is considered to be a sample of the overall graspability. Given a grasp density based on N samples, N new samples are drawn randomly with replacement and a new density is created based on these samples. This procedure is repeated B times and the similarity s of the B re-sampled densities is estimated:

$$s = \frac{1}{B} \sum_{b=1}^{B} BC\left(d_{\mu}\left(x\right), d_{b}\left(x\right)\right) \tag{9}$$

where $BC()$ denotes the Bhattacharyya Coefficient [1] and $d_{\mu}(x)$ is the mean density over the B sets. Given a large value for B, $d(x)$ is used as an approximation of $d_{\mu}(x)$ in equation (9). Note that although s is defined similarly to the variance, the variance approaches zero when s approaches the value of one, which is interpreted as the density being fully converged.

The more the individual kernels of a density overlap, the smaller the variance of the re-sampled densities is expected to be and as a consequence the similarity s will approach 1. Figure 6 shows the convergence of three densities which are based on the same samples but using different widths. Although it is obvious that the larger kernel leads to a faster convergence, it is important to keep in mind that the densities do not converge to the very same density as each larger kernel covers a larger region, even though this does not necessarily reflect the true grasp density. When samples are drawn randomly (and not obtained by the search for a maximum) from the final densities, the average success ratios of the samples have been found to be 9.76% (45mm), 18.18% (30mm) and 30.08%(15mm). Note that these numbers do present

Object	successes	failures	ratio
Cone	166.690	3.331.043	5.0 %
Elongated D	2.898	99.764	2.9 %
Screwdriver	159.947	1.574.709	10.1 %
Mug	19.051	4.127.861	0.46 %

TABLE I: Overview over the number of successful and failing grasps for the individual objects and the success ratio.

results in a very sub-optimal use of grasp densities in which also grasps are tested which are known to have a low success likelihood (i.e., where the grasp density has low values). This can be very useful, when we want to explore grasps corresponding to areas in $SE(3)$ where there are unstable grasps. It is a quality of the grasp density also to represent these kind of areas appropriately. However, one needs to be aware of the trade–off discussed here.

III. Results

The estimated conditional probabilities $P(T_{RBM}(A)$ is successful$|A$ is successful$)$ of the success of the displaced successful grasps for various objects is given in section III-A. Results in the context of the adaption of an anisotropic kernel to these statistics is outlined in section III-B. results on the convergence using this anisotropic kernel are given in section III-C.

A. Statistical Results

A feasible method for an investigation of the results of the statistical investigation are multidimensional histograms where one axis covers the likelihood of a grasp to be successful and each direction of displacement leads to an additional axis. As it is hard to visualize high-dimensional histograms, only two dimensions of displacements will be covered in a single histogram and the mean success likelihoods of the grasps associated with the individual bins are computed.

Given an object that can be grasped at its edge, it is expected that successful grasps can be translated along the edge (see figure 7a where the green marker represents a successful grasp and the orange ones represent grasps we would expect to succeed as well). However, grasps will also be translatable on the two orthogonal directions depending on the finger-width of the gripper (in conjunction with the thickness of the edge) as well as the length of the fingers. Further it is expected that grasps can be rotated around the axis defined by the normal of the surface ending at this edge (illustrated in figure 7b). Figure 7c indicates the associated coordinate system.

The numbers of succeeding and failing grasps for the different objects are listed in table I. Note that the success likelihood is very different for the different objects. The screwdriver is relatively easy to grasp, even a random grasp has a success chance of more

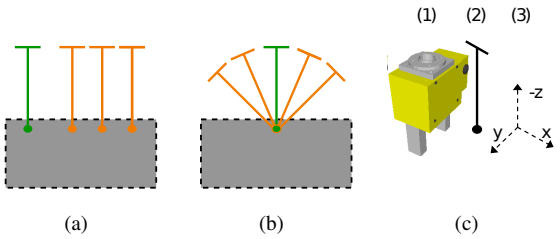


Fig. 7: (a) and (b):Given that the green grasps (visualized in 2D) have been found to be successful and the gray box represents the grasped object, it is expected that the orange grasps would be successful as well. (c) Illustration of the gripper (1), a simplified view (2) and the 3 axes of the associated frame (3).

than 10%. For the 'Elongated D' a random grasp has a probability of success of less than 3% while for the the cone the success likelihood is 5%. For the mug a random grasp has a very low success chance of only 0.5%. Note that for the mug only few grasps will succeed since there are only few graspable positions, and these need to be approached with carefully aligned grasp orientation. For each object an individual grasp density has been created based on the successful grasps (see, e.g., figure 5). Note that the kernel $\mathbf{K}$ does not handle the fact that the gripper is mirror-symmetric around its approach axis. Therefore each sample is used to create the density both unaltered and rotated 180 degrees around the approach-axis of the gripper (the Z-axis, illustrated in figure 7c).

Subsequently each successfully tried grasp is used to generate a set of samples in its vicinity. For each of these samples the grasp density associated with the object is used to estimate the likelihood of graps corresponding to the new sample to be successful. As it is intractable to sample the complete 6D neighborhood for every tried grasp, a kernel is defined at the location of a tried grasp. Subsequently this kernel is sampled. Thereby the entire neighborhood of grasps can be covered, without exhausting it with every single tried grasp.

The average success likelihoods of the transformed grasps relative to the transformation (i.e., $P(T_{RBM}(A)$ is successful$|A$ is successful$)$) for the object 'Elongated D' is shown on the histograms in figure 8. The color of each bin reflects the success-likelihood of the grasps, 0 at the scale indicate 0% success likelihood, 1 indicates 100%.

The histograms in the top row cover the translations — each histograms covers the translations in the X- and Y- axis (horizontal resp. vertical axis on the histogram). Further, each histogram is accumulated over a range of displacements of 16mm in the Z-axis (-24mm to -40mm in the leftmost histogram, 24mm to 40mm in the rightmost). Each histogram in the bottom row covers two axis of rotations. The relation between a grasp and the different axes is shown in

$$P(T_{RBM}(A) \text{ is successful} | A \text{ is successful}), \text{ object not aligned}$$

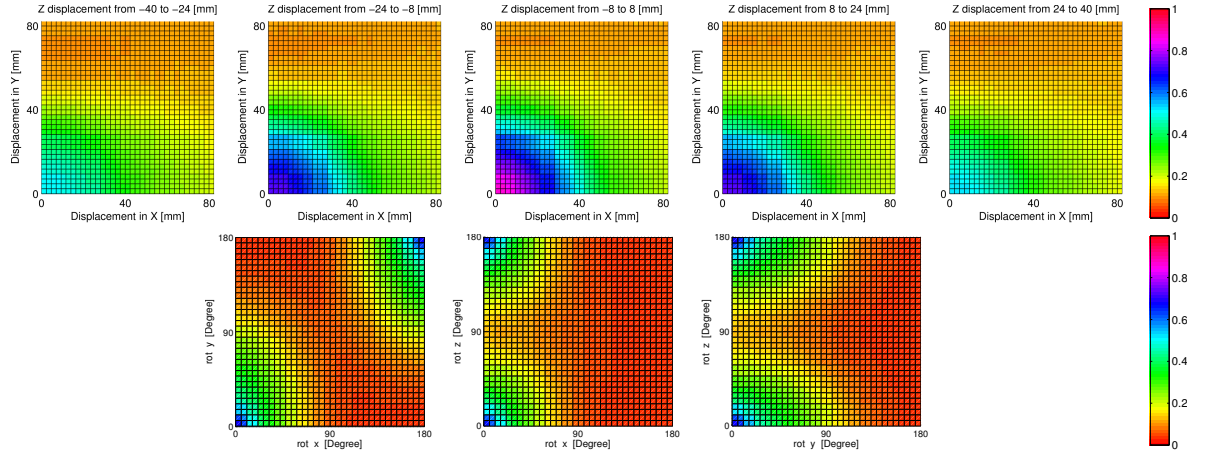


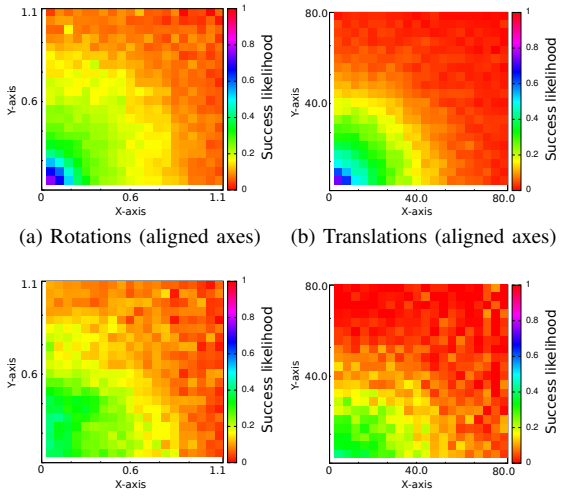
Fig. 8: Results for the object 'Elongated D' for both rotation and translation using the grasp densities with 30mm wide kernels for position.

figure 7c.

Results indicate that successful grasps are more robust towards translation in x– and z–direction and rotations around their y-axis than the two remaining axes. This confirms our expectations visualized in figure 7. Note that when rotations are applied symmetries become explicit (see bottom row in figure 8). When a grasp for instance is rotated 180 degrees around both it's x- and y-axis, the resulting configuration will be identical with the initial one, just mirrored around the Z-axis which is identical with the approach-vector of the tool. Due to symmetry of the gripper the resulting grasp is considered to be identical with the initial one.

However, the results are not as explicit as one might have expected. It can be observed that a wide range of different orientations may still lead to a successful grasp at the very same position. Hence grasps do not have to be aligned with the edge of the object in order to be successful. But when the grasp and the object are not aligned, the translations we apply are not aligned with the object either and will lead to likely to be unsuccessful grasps. This becomes evident when we look at the corresponding statistics when we align the kernel with the main orientation of the object (see figure 9): It can be seen that the anisotropic structure is much more expressive both in translation and orientation. As a consequence of this investigations we can conclude *that it is important to align an anisotropic kernel with the visually extracted edge shape structure.*

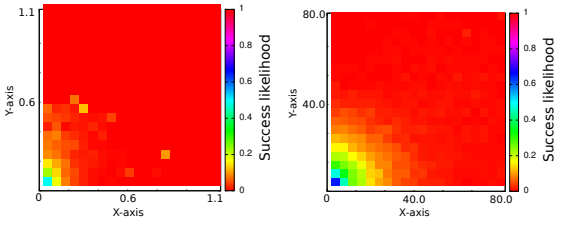
In figure 10 - 12 we see the analog statistics (only the two center sub-figures corresponding to figure 9 and 8 are displayed). As we can see, the results for the screwdriver (figure 10) and the cone (see figure 12) look similar to the object 'Elongated D'. However, the structure of the conditional probabilities is different for the mug (figure 11). The reason is that the mug has only highly curved edges for which only slight translations of successful grasps lead to errors. Here also an alignment to the actual *curved* object shape would be appropriate.



(a) Rotations (aligned axes) (b) Translations (aligned axes)

(c) Rotations (original axes) (d) Translations (original axes)

Fig. 10: Histograms for the screwdriver



(a) Rotations (b) Translations

Fig. 11: Histograms for the mug (not aligned)

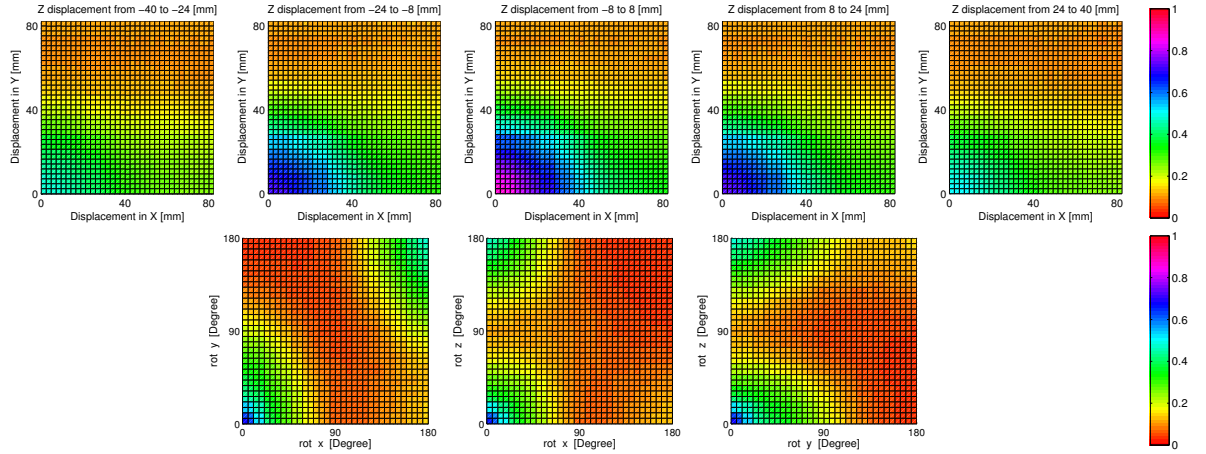
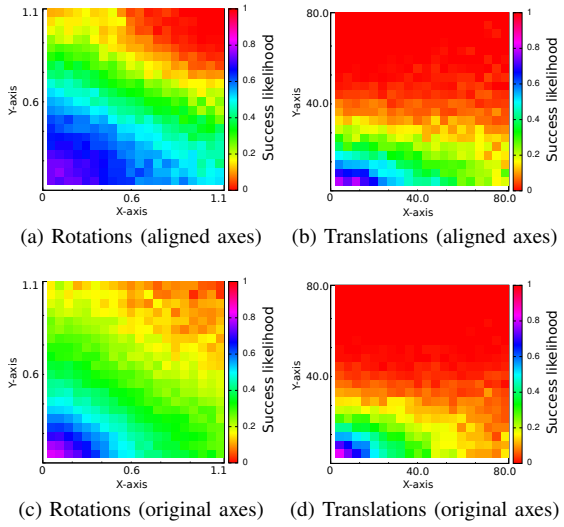


Fig. 9: Results for the object 'Elongated D' when the grasps are translated aligned with the object structure.



(a) Rotations (aligned axes) (b) Translations (aligned axes)

(c) Rotations (original axes) (d) Translations (original original axes)

Fig. 12: Histograms for the cone

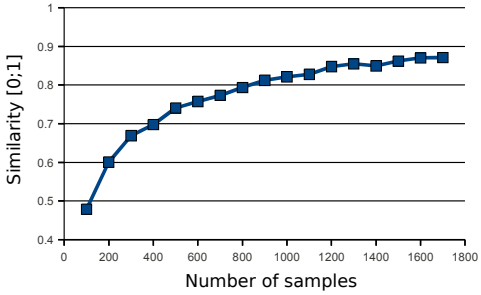


Fig. 13: Comparing the sampled kernel with the full data set used for the histograms for different numbers of samples.

B. Adapted kernel

When using the definition of a kernel given in equation (5) the anistropic kernel is defined as a sum of isotropic kernels. This sum of kernels can directly be obtained by drawing are sufficiently large number of samples from the results that form the basis for the histograms in section III-A. Figure 13 illustrates the similarity between the results used for the histograms and the anisotropic kernel with respect to the number of samples used for the kernel estimation. The results show that a rather high number of samples are required in order to achieve estimate of the anisotropich kernel. In order to limit the computational costs we limit the kernel to consist of 900 samples which leads to a similarity of approximately 0.8, estimated using the Bhattacharyya Coefficient [1].

C. Convergence

The usage of the anisotropic kernel does not imply significant changes to the learning of a grasp density. Independent on the type of the kernel, one kernel is added to the density every time a successful grasp is experienced. When the anisotropic kernel is used all kernels that it consists of are transformed according to the pose of experienced grasp and subsequently the composition of kernels is added to the density. Figure 14 shows a comparison of the convergence of two densities (both are based on the set of grasps learned for the elongated D), one using the anisotropic kernel and one using the isotropic kernel with a similar width. It becomes explicit the density that is using the anisotropic kernel converges significantly faster.

IV. Conclusion and Discussion

In this paper we have investigated the conditional probabilities of grasps in the vicinity of an already performed successful grasp for a number of objects. We found a large degree of structure in these conditional probabilities with large anisotropies. We have used these anisotropies to derive statistically justified kernels for grasp density estimation [3] and we have

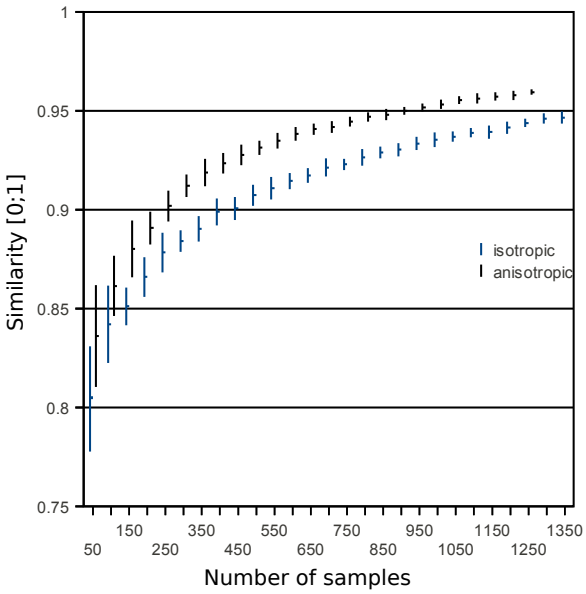


Fig. 14: Comparing convergence of the isotropic kernel and the anisotropic kernel. The vertical bars indicate the standard deviation of the individual similarity measurement.

shown that based on these kernels a faster convergence of grasp densities can be achieved. By that the statistical experience made during grasping can influence the actual learning approach on a meta level. We believe that the efficient use of such statistically derived conditional probabilities is one of the main reasons for successful development of cognitive agents. We have also shown that it is important to align the derived kernels with the actual structure of the object shape: Considering the scenario in [3] where local 3D edge-descriptors of the scene have been used to create a proposal grasp density describing potential grasps. We imagine that the introduction of anisotropic kernels allows us to utilize the edge information further. Rather than generating a proposal density consisting of a vast number of kernels, anisotropic kernels can be fitted to edge segments, thereby covering structural similar regions. As a consequence less samples are needed to formulate a proposal density and less samples are required to sample the entire object.

In future work we will finalize the experiments with the analytically defined anisotropic kernel. Furthermore we aim at comparing the simulation data with real robot data and to derive higher order conditional probabilities associated with more complex feature grasp associations such as coplanar and/or parallel edge and surface structures.

V. Acknowledgments

This work was supported by the EU Cognitive Systems project XPERIENCE (FP7-ICT-270273).

References

[1] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by probability distributions. Bull. Calcutta Math. Soc., 1943. p. 99 - 109.

[2] R. Detry, E. Başeski, N. Krüger, M. Popović, Y. Touati, O. Kroemer, J. Peters, and J. Piater. Learning object-specific grasp affordance densities. In *Int. Conf. on Development and Learning*, 2009.

[3] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning grasp affordance densities. *Paladyn Journal of Behavioral Robotics*, (accepted), 2011.

[4] C. Ferrari and J. Canny. Planning Optimal Grasps. In *IEEE Int. Conf. on Robotics and Automation*, pages 2290–2295, 1992.

[5] R. Fisher. Dispersion on a sphere. *Royal Society of London. Series A, Mathematical and Physical Sciences*, Volume 217 Issue 1130:295–305, 1953.

[6] R. W. Johnson. An Introduction to the Bootstrap. *Journal of the Royal Statistical Society*, 23:49–54, 2001.

[7] J. A. Jørgensen and H. G. Petersen. Usage of simulations to plan stable grasping of unknown objects with a 3-fingered Schunk hand. In *IEEE Int. Conf. on Intelligent RObots and Systems (IROS), Workshop - Robot simulators: available software, scientific applications and future trends*, 2008.

[8] P. C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Science of India*, 12:49–55, 1936.

[9] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, 1986.

LEARNING PEG-IN-HOLE ACTIONS WITH FLEXIBLE OBJECTS

Leon Bodenhagen[1], Andreas R. Fugl[1,2], Morten Willatzen[2], Henrik G. Petersen[1] and Norbert Krüger[1]
[1]*Maersk McKinney Moller Institute, University of Southern Denmark, Campusvej 55, 5230 Odense, Denmark*
[2]*Mads Clausen Institute, University of Southern Denmark, Alsion 2, 6400 Sønderborg, Denmark*
{*lebo, arf, hgp, norbert*}*@mmmi.sdu.dk, willatzen@mci.sdu.dk*

Keywords: Peg-In-Hole, Flexible Objects, Action Learning

Abstract: This paper presents a method for learning Peg-In-Hole actions with flexible objects. To learn the actions we parametrize the entire trajectory by a single point and use Kernel Density Estimation to reflect the different variations of the action and the object characteristics. The object is characterized by its elastic behaviour rather than geometric properties. Thereby an action learned for one object can be transferred to a new object that behaves similarly although it might have different elastic properties, dimensions and geometries. To bootstrap the learning mechanism, the system performs simulated actions and utilizes the detailed information obtained from the simulation environment. Subsequently Peg-In-Hole actions are tested successfully on the real life setup.

1 Introduction

Humans can perform a huge variety of different and apparently simple tasks, but often such tasks are difficult for robots to perform. The Peg-In-Hole problem is one of these tasks and has been studied in numerous works with different perspectives and objectives, often as an example of an assembly task.

One of the aspects investigated is for instance, in addition to the insertion of the peg, the exact alignment of the peg with the hole (Bruyninckx et al., 1995). Assuming that the shape of both the (rigid) peg and the hole is known, the contact forces during the operation can be predicted (Meitinger and Pfeiffer, 1996) and used to optimize the action. More recent approaches often focus on sub-aspects of the classic Peg-In-Hole task. Elastic contacts have for instance been utilized in (Xia et al., 2006) to avoid wedging.

However, to our knowledge only little work has been done with flexible objects in the context of Peg-In-Hole operations or assembly tasks in general (see also (Jiménez, 2011)). In (Villarreal and Asada, 1991) the concept of flexible objects has been used to model finite collision forces between the object and the rim of the hole and thereby aid the motion planning by providing it a "buffer", but in general the shape is considered to stay roughly constant. Path planning with simple, flexible 3D objects like tubes that change their shape during operation are done by (Anshelevich et al., 2000). They model the objects by mass-spring models and perform a random search for the path with the minimal energy. Such an approach is however not feasible when a variety of non-trivial 3D shapes is considered.

In general it is intractable to model and plan the entire action when the deformation of the object has to be considered during the action, therefore this paper investigates an approach that avoids heavy online calculations. Furthermore a classic force-torque sensor can hardly be utilized as any contact will, in addition to measurable forces, cause a deformation of the object - hence standard approaches used for Peg-In-Hole actions with rigid objects cannot be applied.

In this paper we propose a system that learns how to perform the Peg-In-Hole operation with flexible objects (see Figure 1). The learning mechanism has only little prior knowledge about the object; instead the learning utilizes a physical modelling from the elastic properties of the object. The elastic behaviour is derived from calculating the deformation of the bottom surface in the object. By this surface the object is implicitly deformed in the learning stage. This allow us to handle non-trivial 3D shapes in a low-dimensional way. Further a learned action can be transferred to a similar but not necessarily identical object. This leads to a system that can perform in real time as the demand for servoing or online modelling becomes minimized, assuming that most objects in e.g. a production scenario indeed are similar.

The overall system that forms the context for the

action learning is outlined in section 2 and the applied methodology is described in section 3. Section 4 summarizes the experiments that have been done in order to investigate the usability of the suggested approach.

2 System Setup

The overall system, presented in more detail in (Jordt et al., 2011), corresponds to a short production line: Objects are transported by a conveyor belt, a 3D scan of the travelling object provides a 3D triangular mesh of the object. Assuming that the material properties are known, the elastic behaviour of the object is modelled. At the end of the conveyor this knowledge is used to grasp the object. Subsequently an additional action can be performed. The Peg-In-Hole operation is investigated in this paper as it has been considered to be characteristic for many tasks where some sort of object is inserted into a machine in order to be processed.

This paper (in contrast to (Jordt et al., 2011)) focuses primarily on the modelling of deformations as well as the learning of Peg-In-Hole actions. The robot arm with an 1-degree of freedom gripper attached is shown on Figure 1 with a close-up of a Peg-In-Hole operation.

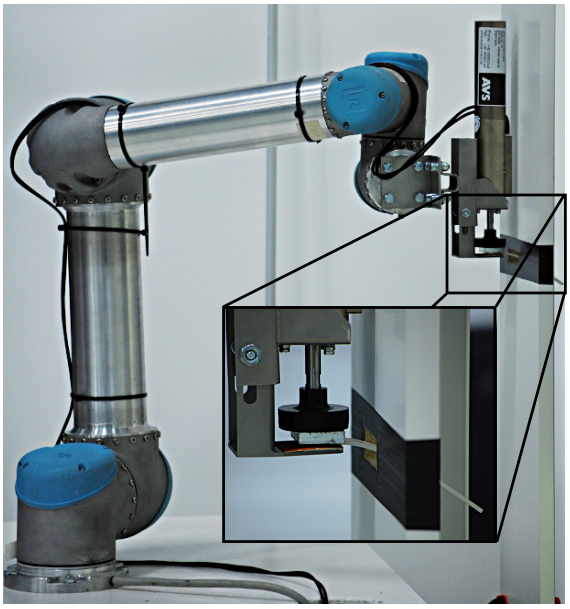


Figure 1: The physical setup used for the experiments.

3 Methods

In the following a detailed overview of the components which this papers focuses on is given. The modelling of the deformations of objects is described in section 3.1, in section 3.2 the physical modelling is condensed into a feature vector and the formalization and learning of actions is defined in section 3.3.

3.1 Deformation Modelling

Deformation modelling in the context of Peg-In-Hole operations, is concerned with modelling the flexible objects in the scene and solving for their behaviour. We restrict the problem to the situation of the peg being substantially more flexible than other objects in the scene. Thus the boundary of the plate, defining the hole for insertion is assumed to be a rigid body, as are the jaws of the robot gripper grasping the peg.

For the flexible peg we want to determine the mechanical response, i.e. how do material points in the peg change as a function of time and external influences (modelled as forces). We assume the elastic parameters such as stiffness and mass density to be available with reasonable accuracy.

In the following, the approach to model deformation for the purpose of learning Peg-In-Hole actions will be outlined.

3.1.1 Deformation description

Let x be a material point in the undeformed object. The object deforms and the new position of the point after deformations are added is x'. The *displacement vector* for some point is thus $u = x' - x$, or in component form:

$$u_i = x'_i - x_i \tag{1}$$

where $i = 1,2,3$ refers to the x,y,z components. The displacement vector is a dense and very general description as it explicitly provides the deformation of every material point in the flexible object. However directly using the deformation vector of material points for the parametrization of general 3D objects becomes prohibitively expensive. For the purpose of reducing the time required for sampling when learning Peg-In-Hole actions with flexible objects, it is crucial to have a sparse but still accurate representation of a deformed surface.

(Samareh et al., 1999) reviewed several shape parametrization techniques, including discrete, polynomial and spline representations. Their goal was to investigate the applicability of the techniques to describe aircraft airfoils with the minimum amount of

parameters. This is important for the purpose of automatic optimization, where the shape of the wing is deformed in small increments to find the best possible aerodynamic design. In this process a large parameters space must be searched, and thus having a small amount of parameters is crucial for the feasibility of the approach.

The parametrization by the discrete approach corresponds to sampling the displacement vector at regular intervals at the boundary. It is the most straightforward method, and can approximate any shape. However as (Samareh et al., 1999) points out, this degree of freedom is rarely useful due to the inherent smoothness of many objects. For instance smooth, curving features will require many discrete points and accordingly the number of parameters can increase to unacceptable sizes.

Parametrization by polynomia and splines on the other hand exploits the smoothness of the original shape. For smooth shapes they will reduce the number of parameters considerably. The non-uniform rational B-spline, NURBS (Piegl and Tiller, 1997), is best suited for handling a large set of shapes, including analytical shapes such as cylinders, cones and scanned unstructured 3D data (Samareh et al., 1999; Bardinet et al., 1995).

As demonstrated by (Jordt et al., 2011) a real-time tracking of a detailed 3D mesh, using depth and colour video from a Kinect camera, can be coupled to a low-dimensional NURBS surface, see Figure 2.

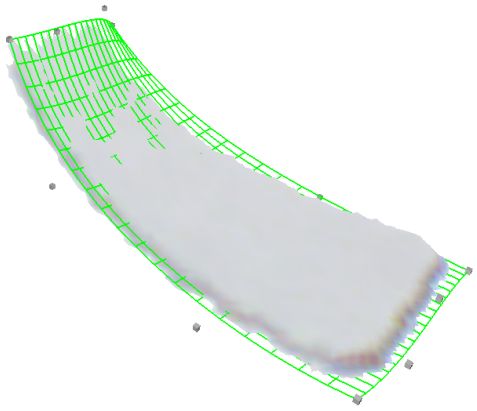


Figure 2: A scanned 3D mesh of an object and its associated NURBS surface (Figure courtesy of (Jordt et al., 2011), with kind permission by the authors)

Similarly we decouple the geometry from the deformation. Only the deformation of the control points in the NURBS surface is used as a a parameter in the learning stage.

The deformation modelling is thus reduced to the problem of finding the deformation for the control points of the NURBS surface. When at some point the whole surface deformed geometry is needed (for instance for collision detection), it is derived from the deformed control points. Having a deformation that models to the NURBS surface also enables easier coupling to the NURBS based deformation tracking.

3.1.2 Choice of model

The Bernoulli-Euler (BE) beam theory has since its development in the 18th century, been a core element in structural engineering. Its' formulation and parameters are readily understandable, and many problems have analytical solutions. It is a simple model however, as it only accounts for the bending moment and lateral displacement of the beam.

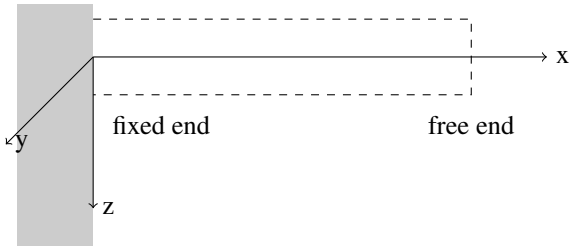


Figure 3: A cantilevered beam.

Several additional models have been developed during the years to improve on the BE model. Most noteworthy of these is the Timoshenko model (Timoshenko, 1921), which takes into the account both rotation inertia and shear deformation.

To account for the additional effects, the Timoshenko model adds a dependent variable to account for the angular displacement and a parameter known as the shape factor. The shape factor is a function of Poisson's ratio for the material, the wave frequency and the shape of the cross section. For the static case, the shape of the cross section is the most dominant effect on the shape factor[1].

The slenderness ratio is the ratio of the beam length to the radius of gyration, calculated as $L/\sqrt{I/A}$. It characterizes the magnitude of different forces involved in the beam equations. In the work of (Seon M. Han, 1999) they recommend the use of the simple BE model for large slenderness ratios ($s > 100$), and the Timoshenko model for smaller ratios where second order effects of rotation and shear become important.

For our present experiments, we target moderately slender objects ($100 < s < 150$). Accordingly we use the BE beam theory.

[1]Poisson's ratio varies for normal materials only between 0 and 0.5 (Landau et al., 1986)

3.1.3 Bernoulli-Euler beam theory

The governing equation for the dynamic BE beam can be formulated as a partial differential equation in the deflection w of the beam

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 w}{\partial x^2}\right) = -\mu\frac{\partial^2 w}{\partial t^2} + q \tag{2}$$

where $w(x,t)$ is the deflection as a function of position x on the beam and time t. E is Young's modulus, I is the second moment of inertia and q the is body load.

Young's modulus, E is a material dependent parameter and represents the stiffness of the material. It may be either measured or derived from tabulated data. For homogeneous materials it is a constant.

The *second moment of inertia*, I is a geometry dependent parameter, quantifying resistance to bending at a given cross section. It is defined as $I = \int_A z^2 dA$, where z is the height of the cross section, being perpendicular to the bending. For a geometry that has a constant cross section (e.g. a simple beam) it is a constant. For the special case of a rectangular cross section with height h and width b, I is equal to $bh^3/12$. This suggests a strong dependence of the thickness of such a beam, to the resulting deformation, i.e. varying the thickness will give the strongest resulting change.

The *body load*, q represents an external force acting upon the beam. It is defined as a force per unit length. Point forces may be modelled with the use of the Dirac delta function.

For the static case of a homogeneous beam with constant cross section, Equation 2 reduces to the ordinary differential equation (ODE)

$$EI\frac{d^4 w}{dx^4} = q(x) \tag{3}$$

where $w(x)$ is the deflection now only as a function of position, and E and I are both constants.

The static beam equation, Equation 3 is a fourth-order ODE. In order to find a unique solution for the deformation profile $w(x)$, four boundary conditions must be prescribed. Assuming that the gripper is placed such that the left end of the object at $x = 0$ is fixed in space (both deflection and slope equal to zero) we have the boundary conditions for the clamped end

$$w|_{x=0} = 0 \quad ; \quad \left.\frac{\partial w}{\partial x}\right|_{x=0} = 0 \tag{4}$$

For the other end of the object at $x = L$, we prescribe the boundary conditions (both the bending moment and the shear force in the beam is zero) corresponding to that this part of the object is free to move

$$\left.\frac{\partial^2 w}{\partial x^2}\right|_{x=L} = 0 \quad ; \quad \left.\frac{\partial^3 w}{\partial x^3}\right|_{x=L} = 0 \tag{5}$$

The ODE Equation 3 together with the boundary conditions Equation 4 and Equation 5 form a boundary value problem. The solution gives the deflection of a fixed-free/cantilevered beam, as depicted on Figure 3.

This boundary value problem, along with the restriction that the load is uniformly distributed i.e. $q(x) = constant$, has the analytical solution to the deflection $w(x)$ of the beam

$$w(x) = \frac{qx^2(6L^2 - 4Lx + x^2)}{24EI} \tag{6}$$

3.2 Object Description

One aim of the action learning is to be able to apply an action learned with one object to another object that behaves similarly. The behaviour of an object is considered to be defined by the deformation that occurs when a specific grasp is applied and the object is affected by gravity - these deformations can be modelled as outlined in section 3.1.

In the following the condensation of the high-dimensional information that is intrinsic to the deformation modelling into a feature vector of lower dimensionality is described. Ideally two different objects, e.g. with different shapes, maps to the same feature vector if they behave identically, such that the same action can be applied.

In order to achieve a feature-vector that is comparable across objects, the NURBS surfaces describing the undeformed object $S_u(u,v)$ and the object in a horizontal orientation, affected by gravity $S_d(u,v)$. The length of the objects is normalized.

The difference between the two surfaces describes how much the object has deformed at the individual locations:

$$\hat{S}(u,v) = S_u(u,v) - S_d(u,v) \tag{7}$$

Based on a regular grid g of size $I \times J$, the deformations are obtained at a set of discrete locations and for a feature vector f:

$$f = \left[\|\hat{S}(g_{00})\|, ..., \|\hat{S}(g_{ij})\|, ..., \|\hat{S}(g_{IJ})\|\right] \tag{8}$$

where g_{ij} refers to the a point of the grid at the position (i,j). An simplified example for the calculation of f is shown in Figure 4. It correspond to a deflecting beam where the deflections can be described by a NURBS curve instead of an entire surface.

3.3 Action Learning

The exact 6D trajectory of a Peg-In-Hole operation depends both on the elastic behaviour of the object, the grasp applied to the object and the shape of the object. However, although the 6D trajectory for instance varies with the size of the object, it might still share similarities with other Peg-In-Hole actions. The parametrization of the action aims to reduce the complexity of the learning problem and eases the transfer of a learned action from one object to another as the object does not need to be identical, but only to share certain properties. The following sections cover the parametrization of Peg-In-Hole actions as well as the structure and strategy for learning them.

3.3.1 Action Parametrization

The Peg-In-Hole action is defined by a trajectory which the robot executes. The endpoint P_1 is known as it is directly in front of the hole. The startpoint P_0 is obtained online utilizing the deformation prediction and ensures that the end of the object is horizontal and in front of the hole (see Figure 5). The trajectory from the start to the endpoint is considered to be approximated by a curve defined by two-dimensional translations and one-dimensional rotations. The points P_0 and P_1 are therefore both points in $\mathbb{R}^2 \times SO(2)$.

The curve $P(t)$ is defined using a rational Bézier-curve (Piegl and Tiller, 1997) based on three control points: the start and endpoint as well as one additional controlpoint which will be obtained by learning:

$$P(t) = P_0 + B(t)(P_1 - P_0) \text{ for } t \in [0;1] \tag{9}$$

with

$$B(t) = \frac{\sum_{i=0}^{n} b_{i,n}(t) P_i w_i}{\sum_{i=0}^{n} b_{i,n}(t) w_i} \tag{10}$$

where $b_{i,n}(t)$ is the Bernstein polynomial with $n = 2$ and P_i refers to the i'th controlpoint for the curve:

$$P_i \in \{\mathbf{0}, cp, \mathbf{1}\} \tag{11}$$

The weights are fixed, $\mathbf{w} = [1,2,1]$, which ensures that the second control point, cp, has an increased impact. Thereby also motions that lead to a significant overshoot can be learned.

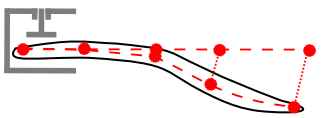


Figure 4: Illustration of the differences between the undeformed mesh (straight dashed line) and the deformed mesh (bent dashed line) used for the feature vector in a 2D case. The resulting feature vector will be 5-dimensional.

Note that the control points do not depend on the scale of the motion or the object (see Equation 9). Therefore a learned control point will lead to meaningful trajectories for any object, although it is not guaranteed that the performed action will be successful.

3.3.2 Action Learning Framework

The set of potentially successful Peg-In-Hole actions is modelled using Kernel Density Estimation (Silverman, 1986). Every time a control point that leads to a successful action has been obtained, it is added to the density d. However, contrary to the situation in (Detry et al., 2011) where grasp affordances are learned for a specific object, we cannot assume the objects to be identical. Therefore a kernel, $\mathbf{K}_{\mu,\sigma}(cp, f)$, which is a compound of two kernels is used: one reflecting the Peg-In-Hole action as such, the other reflecting the object features specified in section 3.2.

$$\mathbf{K}_{\mu,\sigma}(cp, f) = \mathbf{N}^{PiH}_{\mu_p,\sigma_p}(cp)\mathbf{N}^{Object}_{\mu_o,\sigma_o}(f) \tag{12}$$

where $\mathbf{N}^{PiH}$ and $\mathbf{N}^{Object}$ are isotropic multivariate Gaussian kernels located at the mean positions μ_p resp. μ_o and with bandwidth of σ_p resp. σ_o. The density is given by a weighted sum of the m kernels:

$$d(cp, f) = \sum_{i=0}^{m} w_i \mathbf{K}_{\mu_i,\sigma}(cp, f) \tag{13}$$

where the weights w_i ensure that the density integrates to one, hence $\sum_i^m w_i = 1$.

During the learning every controlpoint that leads to a successful action will contribute to the density with one particle. Assuming that an action is either successful or not, all particles of the density have equal weights. Given an uniformly sampled search space, the value of the density at a given point will be proportional to the likelihood of the corresponding action for being successful.

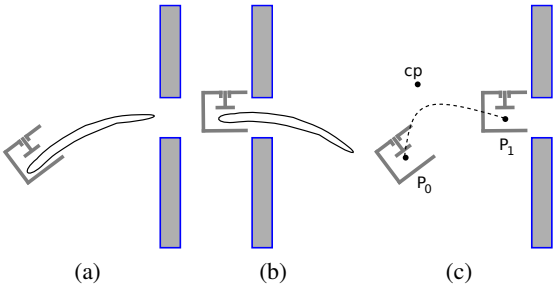


Figure 5: Illustration of (a) starting configuration P_0 and (b) target configuration P_1 for the Peg-In-Hole action. (c) shows a projection of the 3D trajectory based on P_0, P_1 and the controlpoint cp.

Here, we choose the points for the feature vector f as those illustrated in Figure 4. It should be noticed that the point P_0 is scaled with respect to object length (see Figure 4). Thus, two objects with different lengths having the same feature vector then have equivalent shapes and may be handled in the same way except for choosing the appropriate length scaled P_0. Thus, the parameters cp and f covers a given control point and shape for all object lengths.

Assume now that we wish to solve a Peg In Hole action for a hitherto unstudied object. The deflection model is then used to compute the feature vector f_O. Then the control point with the highest probability for success can be obtained by searching for the maximum of the density $d(cp, f_O)$.

3.3.3 Action Learning Strategy

The system is not provided with any prior knowledge to bootstrap the learning strategy. Therefore a 2-step learning mechanism has been considered. First, an exhaustive search on the controlpoints is performed in a simulated environment. As the controlpoints are 3-dimensional it is feasible to explore the space with a reasonable resolution. The outcome of the simulated experiments leads to a density as defined in Equation 13. Examples for the clouds of particles are shown in Figure 7. Finally, the density achieved by simulation can be sampled and evaluated on the real-world setup, leading to a new density.

Utilizing a simulator does not only allow the evaluation of large set of experiments, it also provides a detailed feedback about the performed action. The outcome of an experiment is therefore not only a binary, namely success or failure, but also the minimal clearance c between the object and rim of the hole that is experienced during the individual experiment. A bigger clearance implies that the action is more robust to external disturbances and modelling errors. This fact is reflected by the weights:

$$w_{ij} = \frac{1}{N} \frac{c_{ij}}{\sum_{j=0}^{M} c_{ij}} \tag{14}$$

where c_{ij} is the minimal clearance of the j'th out of M successful experiments with the i'th object, given N objects in total. Thereby the maximum of the density does not only reflect the success likelihood, based on the statistics of the samples, but directly corresponds to the action that is expected to be the most robust in the given situation.

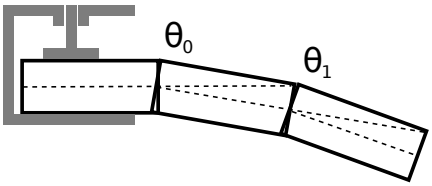


Figure 6: Approximation of a flexible object using a rigid device. Self-collisions within the simplified object model are ignored.

4 Experiments

In the following the simulated experiments used to bootstrap the learning are described in section 4.1. Experiments on the real setup are described in section 4.2.

The test specimens are cuboid pieces of silicone rubber, cut from a sheet of 2.0 mm thickness, into pieces of 15 mm width. The density of the silicone sheet as given by the manufacturer is 1.15 g/cm^3 and the shore A hardness is 60 ± 5 (which corresponds to a Young's modulus of approximately 3.6 MPa).

4.1 Simulated experiments

For the simulation, based on a simulation environment from (Ellekilde and Jørgensen, 2010), a flexible, cuboid object is approximated by a rigid device consisting of a set of consecutive boxes as illustrated in Figure 6. This approximation allows for efficient collision detections as well as clearance calculations - the precision can be controlled easily by adjusting the number of joints in the device. The angles of the joints connecting the boxes are obtained from the object deformation modelling which takes the orientation of the grasped object with respect to gravity into account.

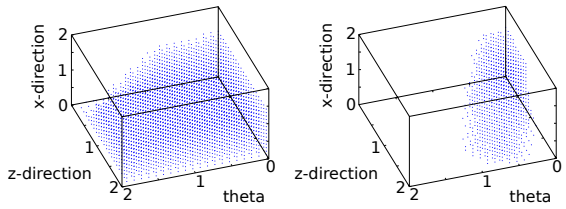
	Minimal clearance [mm]			
Object	collision	0 - 2	2 - 4	4 <
80 mm long	6899	258	288	555
60 mm long	4711	416	524	2349
40 mm long	3418	678	1267	2637

Table 1: Overview over the different outcomes experienced in the simulator.

Simulations have been done for three different objects, testing Peg-In-Hole actions for each object with 8000 controlpoints. The outcomes of the experiments (summarized in Table 1) indicate that it is easier to insert short objects rather than long ones: a higher proportion of all actions succeeded and the average minimal clearance of the succeeding actions is larger.

This has been expected as long objects lead to large deflections and can thus not be inserted by a close to straight-line motion in contrast to short objects.

All controlpoints learned for the short resp. long object are shown in Figure 7. In both cases the solutions form a close to convex area which indicates that the complete density can be approximated with a sparse density consisting of fewer particles, but with larger bandwidths. As the costs of the search for a maximum within the density depend on the number particles that need to be evaluated, densities based on fewer particles ease the implementation of a real-time system.



(a) 40 mm (b) 80 mm

Figure 7: Illustration of the 3D point clouds of the controlpoints that lead to successful actions for the (a) 40 mm long object and (b) the 80 mm long object.

4.2 Real experiments

In the following the validity of the modelled deflections as well as the learned Peg-In-Hole actions are assessed by real-world experiments.

4.2.1 Deformation validation

To validate the modelling, the maximum deformation of each test object have been measured in a separate experimental setup.

	Measured max. deflection [mm]				
Object	O1	O2	O3	O4	Sim.
80 mm	29	28	30	30	33.5
60 mm	10	10	10	11	10.5
40 mm	2	2	3	2	2.5

Table 2: The maximum deflection of the respective test objects. The last column shows the deformation as calculated by the modelling, assuming a shore A hardness of 60. For each object, 4 different orientations have been independently measured.

For the test objects of $80, 60$ and 40 mm, the mean values for the deformations are respectively 29.25, 10.25 and 2.25 mm. Using the tabulated shore A hardness of 60 for the silicone rubber (corresponding Young's modulus 3.6 MPa), the deformation is overestimated. This trend is clear for the larger deformations of the piece 80 mm in length. Using an extrema of the hardness, shore A 65 (Young's modulus 4.4 MPa) the maximum calculated deflection of the piece reduces to 28.3 mm, which is closer to the observed mean of 29.25 mm.

4.2.2 Peg In Hole actions

Based on the results of the simulated experiments, Peg-In-Hole actions with the simulated objects have been evaluated on the real setup. The controlpoints have been obtained by searching the density obtained by simulation for a maximum. The resulting actions have been observed to be successful, the last step of the insertion of the longest object is shown on Figure 1. However manual measurements of the minimal clearance have been done in order to investigate the robustness of the learned actions. Especially for the longest object, the clearance has been observed to be approximately 1 mm (for the 80 mm long object) which is lower than expected according to results in Table 1.

A potential reason for smaller clearance might be caused by alignment errors between the grasped object and the hole as even small errors seam to have a significant effect. Further the most significant difference between measured and expected clearance has occurred for the 80 mm long object, which might be correlated with the fact that the deflection modelling for this object had the bigger error than the others (see Table 2).

5 Future Work

The overall system discussed so far is, as no sensor input is used to correct for modelling errors, an open-loop system. However, when the complete scenario is considered where an object becomes scanned, modelled, grasped and inserted multiple error sources arise. If the object-relative location of the grasp is significantly different than expected, this would have an impact on the modelling and might cause the Peg-In-Hole action to fail.

To counteract potential errors an additional Kinect-camera is introduced, enabling the system to supervise the Peg-In-Hole operation. We foresee that the additional feedback can be used to:

- Improve the deflection modelling over time.
- Correct for inaccuracies during the grasping.
- Correct the starting position of the Peg-In-Hole action.

6 Discussion

In this paper we presented a system to perform Peg-In-Hole action with flexible objects. The system utilizes a physical modelling of the elastic behaviour of the objects and an action learning mechanism based on kernel density estimation. Objects are identified by a distinctive feature vector that enables the system to recognize objects with similar behaviours as known objects. Thereby previously learned actions can be applied to new objects, with similarly behaviour as known ones. This enables the system to perform in real time as the demand for time consuming modelling operations is minimized.

ACKNOWLEDGEMENTS

This work was co-financed by the INTERREG 4 program Syddanmark-Schleswig-K.E.R.N. by EU funds from the European Regional Development Fund. The presented work has also received funding from the EU Seventh Framework Programme under grant agreement no. 270273, Xperience.

REFERENCES

Anshelevich, E., Owens, S., Lamiraux, F., and Kavraki, L. E. (2000). Deformable volumes in path planning applications. In *IEEE International Conference on Robotics and Automation*.

Bardinet, E., Cohen, L. D., and Ayache, N. (1995). A parametric deformable model to fit unstructured 3d data.

Bruyninckx, H., Dutré, S., and Schutter, J. D. (1995). Peg-on-hole: A model based solution to peg and hole alignment. In *International Conference on Robotics and Automation*.

Detry, R., Kraft, D., Kroemer, O., Bodenhagen, L., Peters, J., Krüger, N., and Piater, J. (2011). Learning grasp affordance densities. *Paladyn Journal of Behavioral Robotics*, 2:1–17.

Ellekilde, L.-P. and Jørgensen, J. A. (2010). RobWork: A Flexible Toolbox for Robotics Research and Education. In *International Symposium on Robotics*,

Jiménez, P. (2011). Survey on model-based manipulation planning of deformable objects. *Robotics and Computer-Integrated Manufacturing*, In Press.

Jordt, A., Fugl, A. R., Bodenhagen, L., Willatzen, M., Koch, R., Petersen, H. G., Andersen, K. A., Olsen, M. M., and Krüger, N. (2011). An outline for an intelligent system performing peg-in-hole actions with flexible objects. In *The International Conference on Intelligent Robotics and Applications (Accepted)*.

Landau, L. D., Pitaevskii, L. P., Lifshitz, E. M., and Kosevich, A. M. (1986). *Theory of Elasticity*. Butterworth-Heinemann.

Meitinger, T. and Pfeiffer, F. (1996). The spatial peg-in-hole problem. In *Proceedings of the Second World Automation Congress*, Montpellier, France.

Piegl, L. and Tiller, W. (1997). *The NURBS book*. Springer-Verlag New York, Inc., New York, NY, USA, 2. edition.

Samareh, J. A., Samareh, J. A., and Polynomial, B. B. (1999). A survey of shape parameterization techniques.

Seon M. Han, Heym Benaroya, T. W. (1999). Dynamics of transversely vibrating beams using four engineering theories. *Journal of Sound and Vibration*, 225:935–988.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC.

Timoshenko, S. P. (1921). On the correction for shear of the differential equation for transverse vibrations of prismatic bars. *Philosophical Magazine*, 41:744–746.

Villarreal, A. and Asada, H. (1991). A geometric representation of distributed compliance for the assembly of flexible parts. In *Intematiod Conference on Robotics and Automation*.

Xia, Y., Yin, Y., and Chen, Z. (2006). Dynamic analysis for peg-in-hole assembly with contact deformation. *Internatiol Journal of Advanced Manufacturing Technologies*, 30:118–128.

A Survey of the Ontogeny of Tool Use: from Sensorimotor Experience to Planning

Frank Guerin, Norbert Krüger and Dirk Kraft

***Abstract*—In this paper we review current knowledge on the development of tool use in infants in order to provide relevant information for cognitive developmental roboticists aiming to design artificial systems which develop tool use abilities. This information covers (1) sketching developmental pathways leading to tool use competences, (2) the characterisation of learning and test situations, (3) the crystallisation of six mechanisms underlying the developmental process and finally (4) the formulation of a number of challenges and recommendations for designing systems that are able to exhibit tool use abilities in complex contexts.**

***Index Terms*—IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

This survey paper is targeted at researchers in Artificial Intelligence[1] (AI) who are interested in pursuing a developmental approach[2] to achieve robust object manipulation competence and basic tool use in their systems. The paper presents relevant research from studies in developmental psychology (mostly of human infants). In addition to reporting individual results the paper identifies core mechanisms which we believe to be in operation during the development of tool use in infants; based on these we then present general recommendations which may be useful for those who wish to build artificial systems which exhibit a similar development.

From a roboticist's perspective a central question is how to reach a point in development where planning complex actions with objects is possible. Planning ability relies on a knowledge of planning operators which describe preconditions and postconditions of actions. How the knowledge of such operators develops is a key question for this paper. In the developmental psychology literature there are some constructs which are reasonably close to planning operators, for example the *sensorimotor schema*[3] [2], [6]. A sensorimotor schema gathers together the perceptions and associated actions involved in the performance of a habitual behaviour. The schema represents knowledge generalised from all the experiences of that behaviour. It also includes knowledge about the context in which the behaviour was performed as well as expectations about the effects. This paper will trace the development of sensorimotor schemas from their origins in the first months through to the point where they represent sufficiently abstract knowledge that they can be recruited for planning operations.

We take the *Perception-Action Perspective* on tool use development [5], which sees a continuous trajectory of development from early exploratory interactions with objects and surfaces through to more advanced manipulations including tool use. From this perspective, developments in perceptual and motor skills are potentially very relevant to understanding the abilities which precede tool use and which might serve as foundations for it. Therefore this survey will also devote some attention to these "precursors" to tool use. This is in line with the trend in modern AI to shift the focus away from "high-level" cognitive skills and more towards "lower-level" control of actions in the world [7]. This is not a denial of the existence of high-level skills, but rather a realisation of the need to build them from low-level sensorimotor skills.

Tool use is an interesting phenomenon because it is a very obvious demonstration of intelligence, and it is relatively easy to analyse due to its external manifestation. In addition its ontogeny is particularly interesting because it shows a development from simple sensorimotor behaviours to behaviours exhibiting the hallmarks of advanced cognition. In its simplest forms tool use requires no more than simple context-specific sensorimotor knowledge, such as an expectation about the effect of an action. In its advanced forms tool use requires knowledge of objects, distances, forces, and their interactions, and the ability to manipulate some form of internal representations of these. Furthermore, it is surmised that this knowledge of the physical world may be crucial as a foundation for more advanced cognition [8]. Therefore a study of the development of this foundational knowledge, through tool use, would be an important step towards understanding advanced cognition.

This paper presents concrete examples of infants' behavioural developments as well as some theorising about the mechanisms underlying these. Developmental psychology does not yet have a complete theory of cognitive development[4], therefore roboticists do not have an abstract mathemat-

Frank Guerin is with the University of Aberdeen, Department of Computing Science, Aberdeen AB24 3UE, Scotland. (e-mail: f.guerin@abdn.ac.uk)

Norbert Krüger and Dirk Kraft are with the University of Southern Denmark, Mærsk Mc-Kinney Møller Institute, Campusvej 55, 5230 Odense M, Denmark. (e-mail: {norbert,kraft}@mmmi.sdu.dk)

Manuscript received April ??, 20??; revised January ??, 20??.

[1]This paper will use the term Artificial Intelligence (AI) in the broad sense of the discipline concerned with any type of intelligent information processing in artificial systems, including as subdisciplines cognitive robotics, computer vision, and all areas within computational intelligence.

[2]We will not attempt to argue for the developmental approach here, as that has been done elsewhere (e.g. [1]).

[3]"Sensorimotor schema" comes from Piaget [2], but similar ideas called "sensorimotor process" [3], "skill" [4], or "perception-action routine" [5] are used by other psychologists. We use the term in a broad sense to capture the general idea shared by these works.

[4]There exist interesting considerations of design principles connected to developmental processes (see, e.g., [9]) which can guide design processes. However, a complete theory should account for developments in all domains; it should explain how the same mechanism can develop different types of knowledge depending on the environment it interacts with; it should be detailed enough to allow for the computer modelling of a complete longitudinal developmental sequence.

ical "theory of development" which they could apply to any task (contrast with, e.g., Shannon's mathematical theory of communication [10]). In the absence of such a theory, roboticists who wish to build systems which develop in a similar way to human infants may need to follow the same concrete tasks that human infants do. If a roboticist builds a robot to follow an infant's developmental sequence of increasingly sophisticated tasks, then this increases the chance that the roboticist may discover similar mechanisms of development.

The contribution of this paper is summarised as follows[5]

- It sketches the **development paths** along which several examples of simple tool use may be acquired. This means we describe a sequence of increasingly sophisticated behaviours which lead to some examples of tool use. This is valuable as it often reveals pathways which simplify the acquisition of a behaviour, and intermediate competences which one might not have considered.[6]
- It gives **concrete examples** of simple tool use and their precursors. These simple tasks are good candidate tasks for experimentation with artificial systems because they help to avoid the danger of attempting an overly advanced task (which might force a solution to be coded in a non-developmental way).
- It gives some insight into how **knowledge** (of actions and object relationships) may be **represented**; this comes from our analysis of the sensorimotor basis for such knowledge, which begins with subjective sensorimotor experiences, and gradually becomes generalised to capture more objective knowledge about the effects of certain operations when applied in certain situations.
- It gives some insight into **general mechanisms of sensorimotor development**, and how they apply in the development of tool use (see list of mechanisms towards end of this introductory section).

We capture our view of the ontogeny of tool use in humans with the conceptual diagram of Fig. 1. The diagram shows two parallel tracks of development. On the bottom is the "concrete" track which shows the development of sensorimotor schemas, which are observable in infant behaviour. On the top is the "abstract" track which shows the parallel development of the underlying representations which the infant uses. In this paper, we will describe mainly the "concrete track". For the abstract track we stress that we know very little; psychology is mostly the study of behaviour, and conjectures of cognitive models are quite limited at the present time[7].

The lower (concrete) track shows a directed acyclic graph[8]; each node represents a newly created sensorimotor schema (which corresponds to a new observable behaviour arising at this time). The directed edges of the graph have the meaning "is a necessary precursor"; i.e. the later behaviour builds on the previous one(s). Acquisitions are (mostly) accumulative, e.g. babies suck things less as they get older, but in general they don't forget: Most behaviours that we deal with can still be elicited in older infants. For the concrete track we can categorise the behaviours as belonging to three consecutive and overlapping stages[9] (indicated by the overlapping curves), described as follows.

1) Behaviours Without Objects: This stage starts with the development of a small number of innate behaviour patterns which are in general not linked to any object. The vision and motor system become calibrated, leading to the ability to grasp seen objects, which facilitates the transition to the next stage. In parallel with this on the (upper) abstract track we initially have isolated fragmentary representations which function in limited environmental contexts; these develop and become gradually connected (e.g. across different senses), allowing for transfer of knowledge (see e.g. Sec. V-B5).

2) Behaviours with Single Objects: In this stage accidental events start a linking process between action and object perception; this leads to specification and branching of sensorimotor schemas concerned with single objects; their effects become increasingly predictable. In parallel, extensive training data is generated on concrete object-action experiences; this allows the abstract track to find connections between similar experiences, and so to generalise across them; this constitutes the beginnings of the construction of more general object and affordance representations, which become increasingly task independent (by virtue of the fact that they generalise across multiple tasks). We begin to have a generic and powerful world model which constitutes itself as an *independent* entity which is shared by the different sensorimotor schemes. Unfortunately, this internal world model is only indirectly deducible from the observation of behaviour and constitutes a big challenge to roboticists (see also [13]).

3) Object-Object Behaviours: Further branching and refinement of the schemas continues in this stage, but the new element is that the sensorimotor schemas are extended to deal with relationships among objects, which deliver the basic units for tool use. Because schemas now necessarily deal with relationships among objects, the representation of spatial locations and transforms within space begins to be constructed (abstract track). Object representations become elaborated to integrate parts of objects and different perspectives, as well as physical properties influencing their interaction. In the abstract track we also have some connected fragments of representations which may be reformulated to form a new more general representation subsuming the old versions (the process of *representational redescription* [14]). In addition, at this stage simple examples of planning can already be observed. The schemas are now usable in a wider variety of contexts, and their effects increasingly predictable; therefore it is possible to plan a sequence of actions while still maintaining a high degree of predictability of the effects of such action sequences. These developments (on both tracks) are ongoing and do not stop where our figure stops.

[5]The first two points are purely from behaviour, but the last two are delving into the internal mechanism and hence are more speculative.

[6]This is similar to the way in which the fossil record can reveal pathways along which complex organs developed, whereas in the absence of the fossil evidence the evolutionary development seems difficult to explain.

[7]For example, limited to isolated episodes or aspects development [11].

[8]This graph has a strong similarity with Fischer and Hencke [12, Fig. 2].

[9]Our choice to group things in *three* stages is somewhat arbitrary, as it suits the observations we want to describe; Piaget uses six stages [2], and Fischer uses four [4], for the same period.

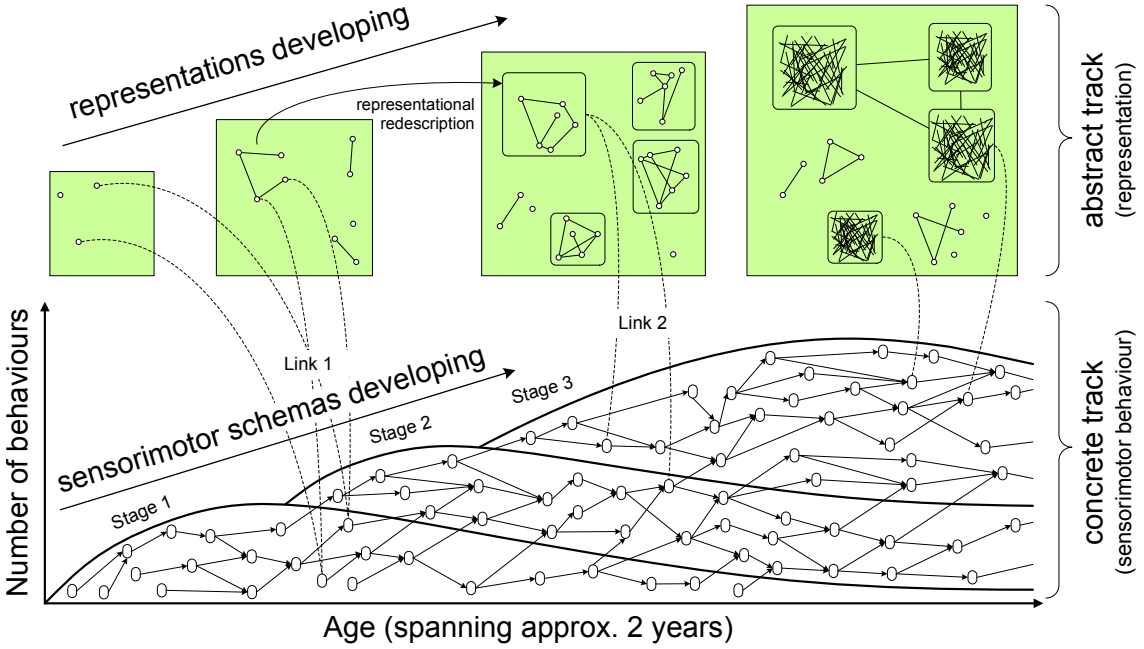


Fig. 1. Conceptual diagram, overviewing infant developments leading to tool use; for explanation see text.

Perhaps the greatest mystery in cognitive development is how abstract general knowledge can come from concrete experiences in specific situations; e.g. infant practice with specific concrete tasks leads to the development of abstract general knowledge of the physical world, such as improving representations of objects and space. We can only give some small insights into this in this paper, and to this end the figure also illustrates (with dashed curves) links between the abstract and concrete tracks; these links are bidirectional. To avoid clutter only a few links are shown, but in reality all representational fragments will be linked to sensorimotor schemas.

In one direction (Link 1 in Fig. 1) representations may be built up (or existing fragmentary representations linked up) from the action of sensorimotor schemas; when a schema acts in a variety of contexts it discovers sensory abstractions which predict its success, and these abstractions are preliminary representations (e.g. representations of shape for grasping (see Sec. IV-A). Such representations can immediately link to actions which can manipulate the represented object or spatial relation. In the other direction (Link 2 in Fig. 1), more advanced schemas make use of the newly formed representations, for example in their description of the context in which a behaviour may be performed, or its effects, or the control policy followed during execution of the schema.

This development process (Fig. 1) allows us to deduce guidelines for how to set up an artificial developing system. In particular the developments on the concrete track are reasonably well studied and observable, allowing us to deduce some of the mechanisms underlying them:

M1 **Repetition**: each sensorimotor schema seeks to repeat itself opportunistically (this explains *play*, Sec. II-B), leading to its own refinement, and also to the accidental discovery of new effects in new situations (such accidental discoveries can subsequently be intentionally exploited).

M2 **Variation and selection**: actions are performed with high variability in order to discover new results, and understand the effects of parameter variations, and later those results that give desired outcomes are selected.

M3 **Differentiation**: sensorimotor schemas are differentiated when an unexpected result is sufficiently interesting to warrant its own specialised schema.

M4 **Decomposition**: a single schema may be broken into a number of sequential chunks so that refinements of the individual parts can take place, as well as flexible reassembly (see e.g. Secs. V-B1,V-D); this can increase the predictability and maturity of the schema.

M5 **Composition**: sensorimotor schemas can be composed to form simple composite sequences, or higher order schemas which control relationships among lower schemas.

M6 **Modularisation**: composite schemas may be initially crudely connected sequences, but can then be refined by repetition, variation and selection, to produce a "smooth atom" [15], which could then be put under control of another (further composition).

We will give concrete descriptions of how these mechanisms are exemplified through different behaviours in the course of the paper. These mechanisms are crucial for the development of planning competences: M1 and M2 are important to increase the predictive power of the schemas, and by means of M3–M6 new schemas (= planning operators) can be generated.

In this paper, we are not addressing the mechanisms underlying the development of the abstract track, but we want to point out that there is a need for a mechanism that synchronises the development of both tracks:

M7 **Representational redescription**: when similarities are noted among a set of sensorimotor schemas, a new more abstract representation can be created, which can refor-

mulate the knowledge captured in the former schemas within a more generic framework.

The largest part of the paper is Sec. V which goes through examples of tool use and precursors, before this it is necessary to first cover some more general preliminaries. Sec. II sets forth our perspective on the problem of tool use and how competence develops. Sec. III overviews various perspectives on cognitive development in order to explain the different psychological approaches and to put the later results in context. Sec. IV gives an overview of sensorimotor schemas (which is the unit of knowledge we will use in analysis in this paper). Sec. V is the main part of the paper and presents the evidence from various behavioural studies. Sec. VI briefly looks at developments on the abstract track. Sec. VII reflects on these results and draws conclusions relevant for cognitive roboticists.

II. The Problem of Tool Use

Tool use is an example of problem solving. It involves selecting the right tool or tools, spatially arranging the right relationships between tools and target objects, and performing the appropriate manipulations to solve the problem. It may be solved by advance planning, or by a simpler trial-and-error in the world (thus we can consider planning as a special case of problem solving). This section defines the problem, and outlines the techniques which human infants (and some animals) seem to apply to simplify the search for solutions.

We distinguish between (i) the general problem solving abilities (such as planning and search techniques) and (ii) the domain specific knowledge of specific actions' preconditions and effects in different situations (i.e. what AI would call planning operators, and which we call sensorimotor schemas); this section focuses on general abilities while we will look at the development of specific abilities in terms of sensorimotor schemas in Sec. V.

Sec. II-A looks at the size of the problem space and how it can be reduced by various techniques. Secs. II-B and II-C look at research on infants' competences in general problem solving and planning, and how these develop. Sec. II-D looks at the role of social learning in problem solving. Sec. II-E sums up what we have learned about the general development of infant problem solving that could be of importance for cognitive roboticists.

A. Managing the Problem Space

A mathematical formulation of the complete problem in a tool using scenario can consider all the degrees of freedom of the actor, and the objects involved (typically a tool and a target object, but possibly other objects as well). The spatial-temporal relation between objects which are to interact is of prime importance; this can be described by a set of relative parameters [16]. Firstly to determine if an object is suitable as a tool for operating on another object to attain a certain goal, the actor needs to monitor, and possibly react to changes in these relative spatial parameters [16]; this implies consideration of the objects' shapes and possible spatial relationships. Secondly in order to use one object as a tool on another, the values of these relative parameters must be appropriately controlled. One must also consider whether these relative parameters [17] (i) must be produced or maintained sequentially or concurrently (sequentially is easier), (ii) require active monitoring for their maintenance, and (iii) are managed by direct contact or through the intermediate action of an object. Furthermore a consideration of required forces and velocities is necessary (which are also parameters).

From this perspective many tool use problems have a very large problem space. In practice there are a number of ways in which the total degrees of freedom are greatly reduced, by dealing with the problem space via smaller manageable subspaces (as exemplified in the next paragraph). This happens because infants (and other animals) tend not to tackle the whole problem space to find a solution, but rather their search is constrained by prior experience, habits, and knowledge. The space reduction methods include (i) *sequencing*, (ii) *stereotypic behaviour*, and (iii) *sensory abstraction*.

Firstly tasks are usually solved by a *sequence of actions*, where each step need only consider (and control) a limited number of degrees of freedom. Consider a capuchin monkey who cracks a nut by first transporting it to a large "anvil" stone, next retrieving a suitable "hammer" stone, and then raising the hammer high to strike the nut [17]. In total there are three objects being put in a relationship, and the number of degrees of freedom is large, but the sequencing of actions leads to a series of smaller problems. It is not necessary to contemplate the relative positions of all three objects; instead one may consider only a pair at a time. To enact a sequence discovered perhaps by chance coordination of components does not require that the whole problem space be considered at one time (*composition*, M5, Sec. I).

Secondly, within any step, the motor actions and the sensory elements considered do not include all those available to the animal, but are constrained by existing sensorimotor "units" which often manifest themselves as *stereotypic behaviours*. An easily identifiable sensorimotor unit is the knowledge related to any habitual behaviour, for example the banging action of an infant. This is what we called a sensorimotor schema in Sec. I (see also Sec. IV), associating perception and action. Infants (and other animals) tend to constrain the actions which they try out on objects to a limited repertoire of stereotypical behaviours [18], even though their motor apparatus has a far greater range of possibilities. This restricts their motion, but they can perform their behaviours with high variation, so there is a distribution of actions associated to each stereotypic behaviour. With the mechanisms of *repetition* (M1) as well as *variation and selection* (M2), these schemas lead to a reasonably constrained exploration of the problem space, while still allowing for less constrained exploration when desired (to provoke new results, and lead to *differentiation*, M3).

Thirdly, *sensory abstractions* constrain the space; for example five-month-old infants use only depth and motion to determine object boundaries (and not colour for example), probably because these have higher ecological validity [19, p. 149]. This perceptual simplification means that such infants face a "smaller" problem than adults in many scenarios, because it is surmised that objects that have lost their depth boundaries are not seen as objects ([20], but see also [21]).

In terms of development these units of knowledge may be partially predetermined by genetics, and/or composed by the organism from other units and fragments (see e.g. Sec. VI). This paper will sketch this development where possible.

We have sketched above *strategies* for facilitating the search in the problem space. The fact that humans (or animals) typically do not consider the whole solution space means that they will often arrive at suboptimal solutions, and this is to be expected in a developmental approach. In many cases when a solution is first assembled the animal will tend to perform component parts in a habitual way (i.e. the way in which those components had been performed before they were recruited to solve the current problem), but over time these may be refined to be more efficient for the task at hand. However an engineer considering all degrees of freedom available to the animal may be able to find a more efficient solution which would not occur to the animal. The cost of the animal's approach is suboptimality, but the benefit is tractability, because the search for a solution may be intractable if all degrees of freedom are considered[10].

The most important fact, for a roboticist, is that the infant system seems to start with mechanisms which ensure rather simple state and action spaces, which then are extended over time. Providing a rather simple initial state space seems also to be reasonable in an artificial developing system and this might even be crucial to make learning and development possible.

B. Planning and Playing

In looking at infant behaviour and development from a "zoomed out" perspective we could see three types of behaviour: reflex, play, and problem solving (these are also overlapping waves). Reflexes seem to happen in early stages of development and serve to bootstrap the development process. Play could be described as an affordance-based activity, where affordances of objects in the environment suggest certain behaviours. There is a close relationship between problem-solving and free-play with infants seamlessly switching between the two; Bruner says [23] "In play, ends are altered to suit means, rather than means being altered to achieve an end held constant, as in problem-solving." Infants sometimes lack the capacity to hold ends in mind, and so the means may take over in some problem solving attempts (e.g. see lifting the barrier in Sec. V-C5). Free play is an effective way for infants to learn about means-end relationships, so it is an important part of the development of planning operators. In this section we focus more on problem solving, but also highlight some connections with play (see Sec. V-C8 for more on play).

Problem solving and planning are very evident in the second year, and to a lesser extent before. These are complex activities requiring task analysis, monitoring of the solution, memory to retain goals and subgoals, organisation of successive attempts, and the use of discovered information to guide further attempts [24]. This search of the space of actions can happen via (at least) three mechanisms[11]: simple forward search, forward search with heuristics, or means end subgoaling (discussed below). The first step in problem solving is to choose a goal which is not immediately attainable. The simplest case is where the goal is seen, such as an out-of-reach object. Alternatively the goal may be recently seen, such as an object that has just been hidden. Sometimes the goal is unseen, but it may be triggered by the sight of something which is often used as an element in a procedure leading to the achievement of that goal; for example an infant sees a coat which triggers the desire to be taken outside. Finally the goal may be internally triggered by a physical need, for example by hunger.

1) Forward Search: This is where actions are tried out in the real world, in an effort to achieve the goal. This does not require any mental simulation of future states or actions. It is required that the infant have the ability to pursue a goal, and in order to avoid exhaustive search of all possibilities, the infant should have some knowledge of how to "use information about the difference between what was achieved and what was intended to guide subsequent activity" [24]. According to studies cited by Willatts, newborns have these abilities, with evidence of some goal directed search with hand to mouth in a limited region. With external objects, goal-directed behaviour may appear as early as three months; this was shown for shaking a mobile hanging above a crib, furthermore these 3-month-olds were able to hold a relatively complex goal in memory (the achievement of a certain amount of shaking in a mobile) [24, see studies cited].

2) Forward Search with Heuristics: Heuristics can give an estimate of how likely a potential candidate action is to lead to the goal. An example heuristic could be the reduction in distance between a desired object and the infant; actions which the infant expects will reduce this distance will be chosen in preference to others. If forward search is used with appropriate heuristics it can lead to a very sophisticated problem solving behaviour, and could explain a great deal of the problem solving observed in infants, even for some of those behaviours which had been thought to be the result of a high-level representation involving mental simulation [24]. For example, the use of a long stick to retrieve an out-of-reach object could be accounted for by a trial and error search, with appropriate heuristics.

Heuristics may also aid in the search for the appropriate parameters for an action (for example the force to be exerted on an object). After varying a parameter the infant may understand if the variation is "going in the right direction". Evidence that such relationships could be deduced comes from [26] who showed that by 15 months (and not before) infants can predict object weight from size (evident from results on grasp development); given that they only have experience of having lifted a certain finite set of objects, they must interpolate for previously unseen objects; this suggests that in general if they have sufficient experience with the effects of some values of a particular parameter or dimension (in perception or motor control) they could interpolate for unseen cases. The kind of knowledge acquired from such interpolation can help greatly in constraining forward search for problem solving, where an action with variable parameters is being used as a means; given a few trials, the relationship between parameters of the action

[10] Apart from tool use, the same strategy of degree of freedom reduction is seen in pure motor control problems (see [22] Sec. 3.4).

[11] These are well known from AI [25, p. 375-416]

being applied, and its effects, could be recognised, and so the range of search can be narrowed considerably. This is an example of the mechanism of *variation and selection* (M2) which allows the relationships between initial conditions and effects to be studied.

3) Means-Ends Behaviour: Here one starts with the goal and searches for a means to achieve it. The simplest form is where a single means action makes the goal action possible. An example is pulling a cloth in order to bring an object resting on it within reach (where grabbing this object is the goal). Simple means-end behaviour has been described by Piaget [2] as emerging about 8 months, and Willatts [24] showed a transition from accidental retrieval to intentional retrieval from 6 to 8 months (see Sec. V-C3). Furthermore, by 9 months it was shown that infants can adjust the means action (cloth pull) as appropriate to the goal, in situations where the goal may be far or near. Willatts argues that the basic ability to perform means-end behaviour is present in the first 6 months, but only appears for manual tasks between 6-8 months because the infant has just acquired new manual skills, and is learning about their effects. In terms of planning operators (or sensorimotor schemas), their preconditions and postconditions are becoming refined via this practice.

What is special about means-end behaviour here is that it generally involves a *composition* (M5) of schemas, one acting on the means object and one on the goal, and so the composition implicitly captures a relationship among these objects, and through practice the infant learns this relationship (i.e. learns situations where the composition works or does not); the pattern here is one of fortuitous success, followed later by understanding (see Sec. V-C3). This type of accidental discovery of relationships among objects could explain the emergence of relational play (i.e. using object-object relationships, see Sec. V-C8) shortly thereafter.

More complex means-end behaviour involves working backwards from the goal to find a series of subgoals which will lead to eventual solution. This requires mental representation of intermediate states (whereas forward search can in principle be done without such representation). Forward search can also be done with mental representation where courses of action are tried out mentally before being tried in the world. As noted above, forward search can also make use of sophisticated heuristics to guide the search, and in observing an infant solving a problem by a sequence of actions it may not be possible to determine if forward search or means-end analysis is being used during that particular episode; Willatts [24] states that he does not know of any empirical way of distinguishing the two alternatives.

4) Affordance-Based Activity within Planning: In addition to simulating forwards (by heuristic search) or backwards (in a hierarchically directed means-end fashion) there is also evidence to suggest that some fragments of solutions in the middle of a possible sequence may be so compelling that children feel obliged to use them. A study of older children (average 32 months) by Cox et al. [16] required them to move a disk (with a duck, a swan, a frog, or a fish painted on top) from the centre of a circular table towards the boundary (which had a trough painted blue for water). In order to move the disk the children were required to use a cane with a hooked shape at the end. They were presented with the cane in a variety of different starting orientations. Despite the fact that the children could have easily swept the disk to the edge of the table in a single motion, on 79% of the trials the children chose to enclose the disk in the hook, and children almost always chose to move the object closer to them. This suggests that the fact that the hook fitted very well with the shape of the disk triggered a fragment of an action sequence that was too compelling for the children to ignore, even though it did not lead to the most efficient solution of the problem. This is an example of where problem solving and affordance based play are not separated.

In summary, for complex problems, it seems plausible that a child may see the scene, and trigger the simulation of many fragmentary sequences of actions; these may be actions from the current state forwards, or from the goal backwards, or parts in the middle of a sequence; these will then be assembled in some sequence which is expected to achieve the goal; this may happen at run-time, or in advance if sufficient knowledge of actions and effects is available. The important message for cognitive roboticists is that there is evidence that basic planning mechanisms are applied at very early stages of development and hence are likely to a substantial degree innately coded but that the library of planning operators is very limited in the beginning, and this explains why not much planning is observed in younger infants.

C. Domain-General Abilities in Problem Solving

The above has descried the main strategies infants seem to use in planning, but has not addressed how the general planning abilities develop over time. Willatts holds that there are no major discontinuous changes in strategy, but he does describe some of the developments in underlying generic cognitive abilities which would lead to improved search in older infants [24]:

- **Memory of what has already been tried**: this has been tested with search tasks, to see that the infant does not return to a location has already been searched. This shows improvement from 14 to 16 months [24].
- **Backtracking** if there has been failure: this has been tested in a task involved nesting cups. Children from 18 to 30 months were unlikely to backtrack to a previous configuration, but from 30 to 42 months there was a significant increase in this behaviour [24].
- **Memory of goals**: younger children may get distracted and forget the goal, the potential depth of their search is therefore limited [24].
- **Organisation of search**: if the order of search is organised systematically it means there is no need to remember what has been tried; for example, infants seem try easy actions first, and later harder ones. The evidence for this organised strategy increases from 12 to 24 months, but it may well be innate, and not very apparent in 12-month-olds simply because they only have simple actions in their repertoire [24].
- **Inhibition of errors** (for example the tendency to repeat a previously successful action, in the wrong situation [27]):

this does seem to improve throughout infancy, but task specific effects are very strong.

It is not clear if the delay in these developments simply reflects limitations of the maturing brain, or if it is important to ensure that a high variation in testing of sensorimotor schemas happens before more sophisticated planning is attempted. In any case, cognitive roboticists might be reminded not to try to trigger complex planning with sensorimotor schemas at too early a stage of development but to focus on the grounding of these schemas.

D. A Brief Note on Social Learning

Most examples of tool use which we cover rely on social learning, either directly or indirectly (for learning precursors to the behaviour). In most socially learnt examples of a skilled behaviour there is an element of imitation and an element of self-exploration. For example, in infant learning of self-feeding with the spoon, the infant initially imitates the behaviour demonstrated by the adult, but the result is quite crude, and the infant shows little understanding of the various components in the sequence of behaviour. Over time (several months), in addition to observing adults, the infant experiments with the constituent parts, and refines the behaviour, eventually producing an effective behaviour. There is also an ongoing interaction between social learning and exploratory learning; there is a limit to how much a learner can advance through learning socially from a master demonstrating a skill, and when the learner does further self-exploratory learning, they subsequently can profit more from the same demonstration (because they now have a greater understanding of the relationships among relevant parameters in the constituent parts).

In this paper we focus on the self-exploration part of the problem rather than the social learning. A great deal of literature exists on social learning in infancy [28], [29], [30], and would warrant a survey of its own. In a robotics scenario it is relatively easy to provide input from a teacher. For example, a human can take hold of the robot's arm and perform an action with it, or the robot can be given a handcrafted example of the correct motion to solve a certain tool use problem in a specific situation. After this the challenge is to make the robot adapt this appropriately in new situations. This requires the robot to develop an understanding of the relevant parameters and relationships among them in the task, as well as appropriate representations, which is what we focus on in this survey.

E. Conclusion for cognitive Roboticists

From the preceding subsections we can draw some important conclusions. First, Sec. II-A suggests that constraints on the complexity of state and action spaces can be designed at the beginning of development, and can help to bootstrap learning and development (and avoid the posing of unreasonably large learning tasks). Secondly, for domain-general aspects we have sketched in Sec. II-B the main strategies (e.g. forward search, means-end) which appear to be in use in early infant planning. It seems that the basic (domain-general) infrastructure for planning is in place relatively early on, but that the library of (domain specific) planning operators is relatively empty, so that not much planning will be observed; this library becomes filled during ongoing development by increasingly accurate planning operators developing from the sensorimotor schemas. Finally, the domain-general aspects of development include the gradually developing abilities listed in Sec. II-C (for example memory); it is possible that following such a schedule of development is advantageous so that the younger infant is presented with simpler more manageable problems (much like degrees of freedom are constrained in early tool use, see Sec. II-A).

Based on this, planning would seem to be relatively easy to emulate in a robot as it matches techniques which are already mature in AI. However, a proper grounding of planning operators in sensorimotor experience seems to require a long development with a great deal of experiences of the causes and effects of sensorimotor schemas, which must be learnt by testing them in many different contexts (by playing). This poses a real challenge to roboticists because it requires *a huge amount* of *meaningful* experiences; this is still difficult to achieve by means of real robots nowadays due to unstable hardware and limited sensors, and inadequate representational structures for interpreting and assimilating the data. The remainder of this paper will mainly focus on the development of these (domain specific) planning operators (i.e., Figure 1 lower track).

III. Perspectives on Development

This section overviews different psychological approaches to understanding infant development. Sec. III-A looks at maturation and learning during the process of development; Sec. III-B overviews Piaget's theory, which is probably the most well-known theory of cognitive development; Sec. III-C looks at Siegler's overlapping waves theory (which we have already borrowed for our three "stages" in Fig. 1); Sec. III-D looks at the more recently popular dynamic systems perspective on development; finally we look at how a consistent picture of development could be found in these theories.

A. Development: Maturation and Learning

Development includes both maturation and learning. Maturation is a change due to biological growth (or aging) in the organism without the need for environmental influences [31, p. 3], e.g. growth of certain centres in the brain. Learning is a change due to information processing; for example a change in the organism's competence resulting from the processing of information from the environment. An extreme nativist viewpoint would posit that all development is due to maturation, with new brain structures unfolding according to a pre-set plan hardwired by the genome. An extreme empiricist viewpoint would posit that all development is due to learning, with new mental structures being constructed due to the processing of new information from the environment (i.e. the software is changing but not the hardware). Contemporary viewpoints lie between the extremes. The evidence from the literature suggests a very complex bidirectional interaction between physical changes (in brain and body) and mental changes due to learning [32]. The impact of changes in the body has been studied in the development of locomotion [32], and

body changes must also pose problems for infants learning to use tools, but we know of no studies addressing this. In this paper we cannot address brain or body changes in any detail since we are mainly concerned with the description of the observable development of sensorimotor schemas. The issue of interaction between innate structure and learning is only indirectly observable (as with the interaction of the concrete and abstract track as indicated in Fig. 1). This problem is also fundamental for roboticists since they need to determine the prior structures of the systems they design.

There is quite some research on the innateness of certain kinds of knowledge in neurophysiology (see, e.g., [33], [34]) and developmental psychology (see, e.g., [19]) which allows for postulates on reasonable innate structures in robot systems (see also [35]). In Sec. V-A, we in particular point to a number of innate behaviours that are used to bootstrap the developmental process (Sec. V-B); we have also given evidence for a certain degree of innate machinery for planning in Sec. II-B.

B. Piagetian Schema Development

Piaget's theory is called constructivism and is based around the idea of the infant gradually building up knowledge structures as he/she interacts with the environment. On the nature/nurture spectrum it is closer to nurture (i.e. the empiricist viewpoint in Sec. III-A). Piaget uses the *sensorimotor schema* (see Sec. I) as the unit of knowledge. Piaget defined six sequential stages during sensorimotor development (approx 0-2 years) [2], [20], with a qualitative difference between the sensorimotor schemas in use in each stage. Piaget's early stages roughly map to the three stages we outlined in Sec. I.

Piaget's Stages I-II roughly correspond to our Stage 1 (Sec. V-A), including reflexes such as sucking and grasping, as well as integrating different modalities (auditory, tactile, visual), and mastering reaching to grasp. Each behaviour is associated with its own global schema which generalises from experiences where the action happens, and recognises the situations where the action is triggered, and the expectation of what sensory impressions arise while the action is in progress.

Piaget's Stage III roughly corresponds to our Stage 2 (Sec. V-B), and involves repeating results fortuitously discovered with objects in the environment, such as shaking a rattle. During this stage there is a rapid growth in the number of schemas in the infant's repertoire, as new schemas are *differentiated* (M3) from previous ones in order to repeat interesting discoveries (e.g., squeezing, shaking, striking, scraping, rubbing, and pulling).

Piaget's Stages IV-V roughly correspond to our Stage 3 (Sec. V-C); means-end sequences of actions are performed. For example, the infant will intentionally displace an obstacle in order to retrieve a desirable object which is visible behind it. This requires two distinct sensorimotor schemas; one for the means action (displace the obstacle), and one for the end (grab the desired object). This implies that the sensorimotor schemas must now incorporate relatively advanced knowledge of the world; they must capture the effect of an action on the relationships between objects (for example the relationship "in front"). Schemas are also intentionally varied (Piaget's Stage V) so that the relationships between initial conditions and effects can be studied. Piaget's Stage VI (roughly from eighteen to twenty four months) involves internal representation of objects, actions, and effects; this gives rise to covert planning (though Willatts is sceptical and sees a more continuous development in planning abilities [24]).

Through all this progression there is a gradual increase in the abstractness and objectiveness of the knowledge captured by sensorimotor schemas; earlier schemas capture subjective knowledge locked in particular contexts, while later schemas abstract away from these contexts and capture knowledge about relationships between objects and actions in the world.

Piaget's stages do not have crisp boundaries between them, and some behaviours are intermediate; the stages also have significant overlap, so that a child who acquires his first Stage V behaviours will also be spending a significant proportion of his time engaging in behaviours belonging to earlier stages. However the sequential ordering is strict, i.e. a child who exhibits behaviour from stage n must have previously exhibited some behaviours from stage $n-1$.

In Piaget's theory development happens either through the modification of individual schemas or the relationships between them. Within each of his six sensorimotor stages schemas individually develop; by being executed in varied situations they refine the motor action of the schema, and also refine their knowledge of the various effects produced in various contexts. Transitions between stages are explained through coordinations among schemas. For example, means-end behaviour emerges in his fourth sensorimotor stage, and this development is explained as a process of coordination between the schemas of previous stages. For example, the schema of hitting an obstacle (means) could be coordinated with the schema of grabbing an object (end) in order to remove an obstacle to prehension.

In the last few decades a great deal has been written about where Piaget was right and wrong; Siegler gives a good brief account [36]. In summary, sometimes Piaget overestimated infants' abilities, and sometimes he underestimated them. However possibly the biggest problem from a computational point of view is simply the vagueness of his theory; it gives a rough sketch of how the development happens, but leaves the mechanism of development very underspecified. Despite all the criticisms, Piaget's theory remains one of the few attempts to explain the whole of development, and quite probably is a reasonable sketch of the outline of how the mechanism of development works.

C. Overlapping Waves Theory

Siegler's "overlapping waves" theory of development [37, see p. 7] holds that at a particular age a child will have a number of different strategies for tackling a problem (for example this could be ways of approaching a particular tool use problem); these different ways of thinking are all active at the same time, and may give rise to different conclusions, thus explaining how a child may approach the same problem in different ways on successive days. The different ways of thinking continue to compete with each other over long timescales (e.g.

several months); with development there are gradual changes whereby more successful ways of thinking become used more frequently, and others are used less frequently.

When compared with a logical/adult approach, this seems to be a "sloppy" way of thinking; for example, if one strategy is clearly leading to failure and another to success, it would seem logical to abandon the first; however, children tend to continue using "wrong" strategies (albeit less frequently) for some time. It is possible that this approach leads to increased robustness, because typical interactions in real world situations have many uncontrolled variables and do not give such clear cut results as a science experiment would. In such situations it may make sense not to abandon any alternative for quite some time, so that there are always alternative strategies to fall back on if the one that first appeared promising eventually proves not to be. Furthermore, it has been shown that children who exhibit more varied ways of thinking learn more from training [38], and more generally, variability in psychological development may play the same critical role that it plays in evolutionary development. This is reflected in the overlapping waves drawn on the lower part of Fig. 1, and these overlapping waves apply equally well to the representational redescriptions in the upper part of the diagram; i.e., older (more context specific) representations will not be immediately retired when newer (more generic) representations come online; the alternative representations will continue to operate in parallel for some time, with one or the other being used depending on the task.

Siegler broadly agrees with Piaget's constructivist theory, but he also highlights the importance of aspects which might be neglected by an excessive focus on constructivism, for example the acquisition of associative knowledge (learnt in specific contexts) or more generally, the issue of knowledge retrieval processes [38]. Siegler points out that it may not make sense to ask "whether children 'have' a concept or strategy or theory at a given age"; instead it may make more sense to investigate "the set of conceptualizations and strategies and theories that children know and the mechanisms by which they choose among them" [38].

D. Dynamic Systems Approach

Piaget gives the impression of a rational infant that will take sensible actions if he/she has the relevant knowledge. The dynamic systems view [32] attempts to explain behaviour at a lower level, via the activation and interaction of various low level processes such as perceiving, moving, and remembering; the eventual behaviour observed is explained in terms of these processes, and may not always appear rational from a more global perspective. This can lead to different conclusions being drawn from behavioural studies: for example, according to Piaget's view, if an infant knows where a hidden object is, then the infant can be expected to attempt to retrieve it from there; however, in the dynamic systems perspective, an infant may reach to the wrong location because of an inability to suppress a response performed earlier [27], or simply because some alternative action has a higher activation (even though the infant might at the same time have an expectation of perceiving the object in the new location). The folk psychology concept of "knowledge" is at too coarse a grain for dynamic systems explanations, so that the question of whether or not an infant really "knows" something (e.g. the location of an object) is not meaningful; the behaviour emerging from the infant's lower level processes may seem to demonstrate knowledge under some circumstances and not others (i.e. context dependent). This relatively new approach to understanding development helps to explain earlier observations which often noted infants' considerable difficulty in inhibiting "obvious" actions, or actions in progress [15]. It also could explain some of Siegler's observations of the context specificity of knowledge, and the switching between different strategies in different circumstances.

E. Conclusion on Developmental Theories

The theories sketched above are not entirely consistent on all details, but it is possible to find a consistent theory which incorporates their major aspects, with some adjustments. From Piaget's theory we can take the notion of sensorimotor schemas, and a mechanism of development which builds new schemas by operations such as differentiation or composition of old schemas (see mechanisms M1–M6). The overlapping waves theory can be accommodated by ensuring that older behaviours will not be replaced at once when newer more sophisticated behaviours develop; instead both will continue in parallel, and may be elicited in different contexts. The dynamic systems approach impresses on us the necessity to model at a fairly low level, so that sensorimotor schemas may be quite context specific, and triggered in certain situations, without a global overview ensuring consistency and rationality in behaviour. This means that developments to new "stages" do not happen all at once, but include a protracted phase of intermediate behaviours where behaviours in some domains are more advanced than others. Abstract domain-general knowledge and representations may be very slow to arise. The brief review above also supports some of the mechanisms we identified in Sec. I as underlying development. For example, the mechanism of *variation and selection* (M2) is very evident, and is believed to be a primary mechanism in development both at low levels such as learning motor synergies [39] and also at higher levels in selecting which strategies to use [38].

IV. Sensorimotor Schemas (Background)

Throughout cognitive development there is an ongoing process of acquiring knowledge; there must exist some organisation for this knowledge. One possible unit of organisation is the sensorimotor schema (as discussed in Sec. I). It includes knowledge about the context in which the behaviour was performed as well as expectations about the effects. In this way it is somewhat similar to an AI planning operator because it gathers together the preconditions and postconditions associated with an action. We have seen in Sec. II-B that much of what is happening during an infant's development towards planning abilities is the *acquisition* of the planning operators; this acquisition process is a major challenge to model since it requires the grounding of sensorimotor schemas in sensorimotor experience by means of the mechanism M1–M6 as well as

the linking to the abstract track by ongoing representational redescriptions (M7). In this paper we are taking the perspective of someone who wants to build an artificial system which develops the capability to plan with simple tools, and for this reason we will trace the development of sensorimotor schemas in our analysis of example behaviours. This section will briefly overview how the sensorimotor schema has been used in psychology and AI.

A. Sensorimotor schemas in developmental psychology

Piaget's theory [2] uses the sensorimotor schema as the unit of knowledge. In this section we look at other related works to show that the sensorimotor schema is not confined to Piaget, but is consistent with ideas in non-Piagetian work, and therefore it is a valid concept to use in our analysis; we also look at some shortcomings of the schema if it is to be treated as the *only* unit of knowledge, and we describe how we can accommodate these extra aspects in our analysis.

The following four post-Piagetian works are consistent with the idea of sensorimotor schemas: (i) There is a close relationship between sensorimotor schemas and the Gibsonian notion of affordances [40]. The visual perception of a handle of a cup for example can be associated with the action of grasping it and the effect of having a stable grasp. (ii) There is evidence that humans do learn action-effect associations which are bi-directional (i.e. the action triggers an expectation of the effect, and the effect triggers recall of the action) and can be used for planning goal-directed actions [41]. (iii) The perception-action routines of Lockman [5] are also very similar to the idea of sensorimotor schemas. (iv) Neuroscientific evidence from monkeys shows that object shape is coded in a motor area of the brain which is involved mostly in the control of hand movements [42]; the authors of this study conclude that "every time an object is presented, its visual features are automatically (regardless of any intention to move) 'translated' into a potential motor action. This potential action describes the pragmatic physical properties of the objects." This lends credibility to the idea of a sensorimotor schema as a unit of knowledge which links a particular perception with an appropriate action.

The sensorimotor schema however always lumps together issues of perceptual, motor, and cognitive development, and in this it is not always compatible with contemporary views. For example, in Piaget's view perception tends to be built up by experience with acting in various contexts. While this has clearly been shown to be the case in an experiment with cat locomotion for example [43], there is less evidence to support all the cases in which Piaget held that the same process occurs. A classic example is in the case of the means-end action of retrieving a hidden object. Piaget held that it was through experiences with acting on objects in relationships such as "in front" that the perceptual competence and representational competence to understand about hidden objects was constructed. Contemporary views hold that many perceptual competences may be more independent from action competence, and that in many cases perceptual competence might come first [19, p. 247,260]. Nevertheless, even if perceptual competence does lead, there may still be a place for action to help with the interpretation of that perception [31, p. 176]. It seems that the idea of sensorimotor schemas is not completely invalidated by later results, but the schema is not the only unit of knowledge and may need to be complemented with pure perceptual or motor competences which may mature according to some internal developmental processes (for example the onset of stereoscopic depth perception [19, p. 96], or the arrival of stereotyped behaviours (see Sec. V-A)); once they do become available it seems plausible that sensorimotor schemas may again come into play to integrate them [44, p. 148]. Fig. 1 (lower part) shows some nodes which do not have precursors, and these correspond to sensorimotor schemas that integrate newly matured perceptual or motor competences.

The use of sensorimotor schemas in analysis does not necessarily mean that we need to bring along all the Piagetian baggage which many have criticised.[12]

It is not clear if the sensorimotor schema can also account for high-level representations of objects, for example 3D models that can be manipulated mentally. Knowledge of objects, which is abstracted from any particular action, would seem to be a prerequisite for the more advanced type of planning. In particular the fact that older children and adults do have excellent generalisation abilities across domains indicates that representations which are independent of particular tasks must arise during development, however the developmental evidence suggests this is a long and protracted process. This is what is illustrated in the upper (abstract) track of Fig. 1.

B. Sensorimotor schemas in robotics/AI

A number of works in AI and robotics have borrowed the idea of sensorimotor schemas, or very similar structures. STRIPS-like planning operators [45] bear some similarity to schemas and have been used to do planning in closed and deterministic worlds. Every action has clearly defined binary pre- and post-conditions, everything is assumed to be observable and only the agent's actions can change the state space in which operations take place. STRIPS-like planning operators (and later extensions [46]) have been used successfully in restricted domains but the major problem is the reliance on human programmers to predefine all operators. Neither the actual action execution by the robot (which can highly vary with the scene context) nor the pre- and post conditions are subject to any learning. As a consequence they require a completely designed world; the limits of this approach have been accurately pinpointed by Brooks [7] and Sutton [47].

There have been a number of works explicitly modelling Piagetian sensorimotor schemas ([48], [49], [50], [51]). These all have three-part schemas of the form context/action/result. Unlike the STRIPS planning these especially focus on allowing schemas to be learnt autonomously from experience,

[12]E.g. Fischer [4] says Piaget's schema is too general, because according to Piaget's theory two skills using the same schema should happen at the same time, and this cannot explain the phenomenon of horizontal décalage (for example that conservation of liquid is acquired long before conservation of mass). This can be circumvented simply by making schemas less general. We may have context specific schemas for each skill, which need not arise together (although some process such as Representational Redescription [14] may later recognise the similarity).

and also to facilitate the *composition* of schemas (M5), and the construction of higher order schemas from lower order ones (see esp. Chaput [49]). These examples mostly work in simplified simulations, and none have attempted tool use.

The concept of affordances [40] has been used to describe behaviours which are triggered by scene or object properties. AI implementations inspired by Gibsonian affordances arrive at knowledge structures very similar to the implementations of Piagetian schemas [52]. Modayil and Kuipers show in [53] how the effects of a physical robot's actions on its perception can be learned. Fitzpatrick and Metta [54] learned affordances for pushing objects and made initial steps towards learning categories (e.g., roll-able). Stoytchev [55] is able to learn tool affordances for a set of tools by correlating the tool used, its movement and the effects on other objects in the scene. Hart and Grupen [56] show how generalisable control programs can be learned and how they can evolve from one task to the next. These generalisable control programs could be understood as the action core of a sensorimotor schema.

The major challenge in formalising sensorimotor schemas is probably to capture the dynamic properties of its actual acquisition process. This acquisition requires us to (i) ground the sensorimotor schemas in sensorimotor experience (meaning to learn suitable pre- and postconditions as well as associated success likelihoods) by means of the mechanisms M1 and M2, (ii) define processes that lead to the creation of new sensorimotor schemas by means of the mechanisms M3–M6, and (iii) integrate the sensorimotor schemas into the dynamically changing abstract world knowledge by means of representational redescription (M7). This in particular requires the performance of a huge amount of meaningful actions providing the required experience to feed the developmental process in a complex cognitive architecture. This architecture is likely to be equipped with a not insignificant degree of innate structure such as innate behaviours and a rudimentary planning machinery (see Sec. III-A) based on which the still to be acquired planning operators develop from initial sensorimotor schemas. In this context, we recently introduced the concept of so called "object-action complexes" (OACs) [57] as one possibility to formalise the complex acquisition process associated to sensorimotor schema.

In this paper we give evidence that the development and enriching of sensorimotor schemas can be directly linked to the development of tool use, and hence provides a means to tackle the same problem in robotics. Insight into the development of sensorimotor schemas in infants hence can also guide the design of artificial agents which learn to use tools.

V. The Behavioural Studies (Concrete Track)

[in this section: NK and DK to add links to robotics work that is close to the abilities described in infants]

This section looks at behavioural developments in the first two years which we believe to be relevant to the development of tool use. The section is organised roughly in order of developments which build on each other, including the supposed precursors to tool use, and simple examples of tool use of increasing complexity. These behaviours are summarised in Fig. 1 where there are three overlapping waves for the three types of behaviour: behaviours without objects (Sec. V-A), behaviours with single objects (Sec. V-B), and object-object behaviours (Sec. V-C). These subsections cover the precursors to tool use, and then Sec. V-D takes an in depth look at one particular example of tool use which is common in the second year of infancy: self-feeding with a spoon. Fig. 2 graphs the individual behavioural developments covered in this section.

A. Behaviours without objects

Here we analyse some typical behaviour patterns of infants which do not require manual contact with objects or surfaces. Firstly there are a number of "reflexes" such as reaching, or rooting for the breast; however von Hofsten [59] cites evidence showing that these and other examples of supposed reflexes do not in fact share the expected properties of reflexes (e.g. elicited, and automatic) and in fact turn out to be under voluntary control. He states that as with other mammals it should not be surprising to find sophisticated prestructured actions in human neonates. Secondly there are "rhythmical stereotypical behaviours" which Thelen describes as being more complex than reflexes, but less variable and flexible than full voluntary behaviour [18]. This lack of variability may be an example of the strategy of initially reducing the degrees of freedom of the motor control problem (see Sec. II-A). Thelen [60] observed infants longitudinally during the first year, and recorded all rhythmical stereotypical behaviours; this meant any movement that was repeated at least three times at regular short intervals of about a second or less. Forty seven distinct behaviours were observed, appearing at different times. We will describe eight of these, involving arms, hands and fingers, which seem most relevant as precursors to tool use (rather than leg movements, or whole body movements etc.). The numbers in parentheses describe the percentage of sampled infants who exhibited the behaviour.

1) Arm wave (100%): a rapid flapping of the arm vertically from the shoulder. This leads to surface slapping behaviour (Sec. V-B6), and also waving of objects and banging them on surfaces (Sec. V-C1).
2) Finger flex (100%); flexion and extension of all four fingers simultaneously, and often the thumb. This probably leads to exploratory behaviours with objects (Sec. V-B5).

3,4) Hand rotate (90%) and flex (80%): a rhythmic rotation, bending and extending of the wrist. This is subsequently performed with objects; possibly it is used in object exploration (Sec. V-B5).

5) Clap hands together (75%) (which Thelen calls Pat-a-cake): This later leads to stereotyped banging objects together (Sec. V-C8) (85%).

6,7) Arm fly together (20%) and Plucking (15%): these were similar to clapping, but the hands were not extended to slap palms together; hands were brought together and then thrown apart; these may also be precursors to banging objects together (Sec. V-C8).

8) Finger rotate (15%): similar to "the movement used in turning a large dial, where the fingers are rotated slightly outward" [60]; this may lead to rotation of lids/dials.

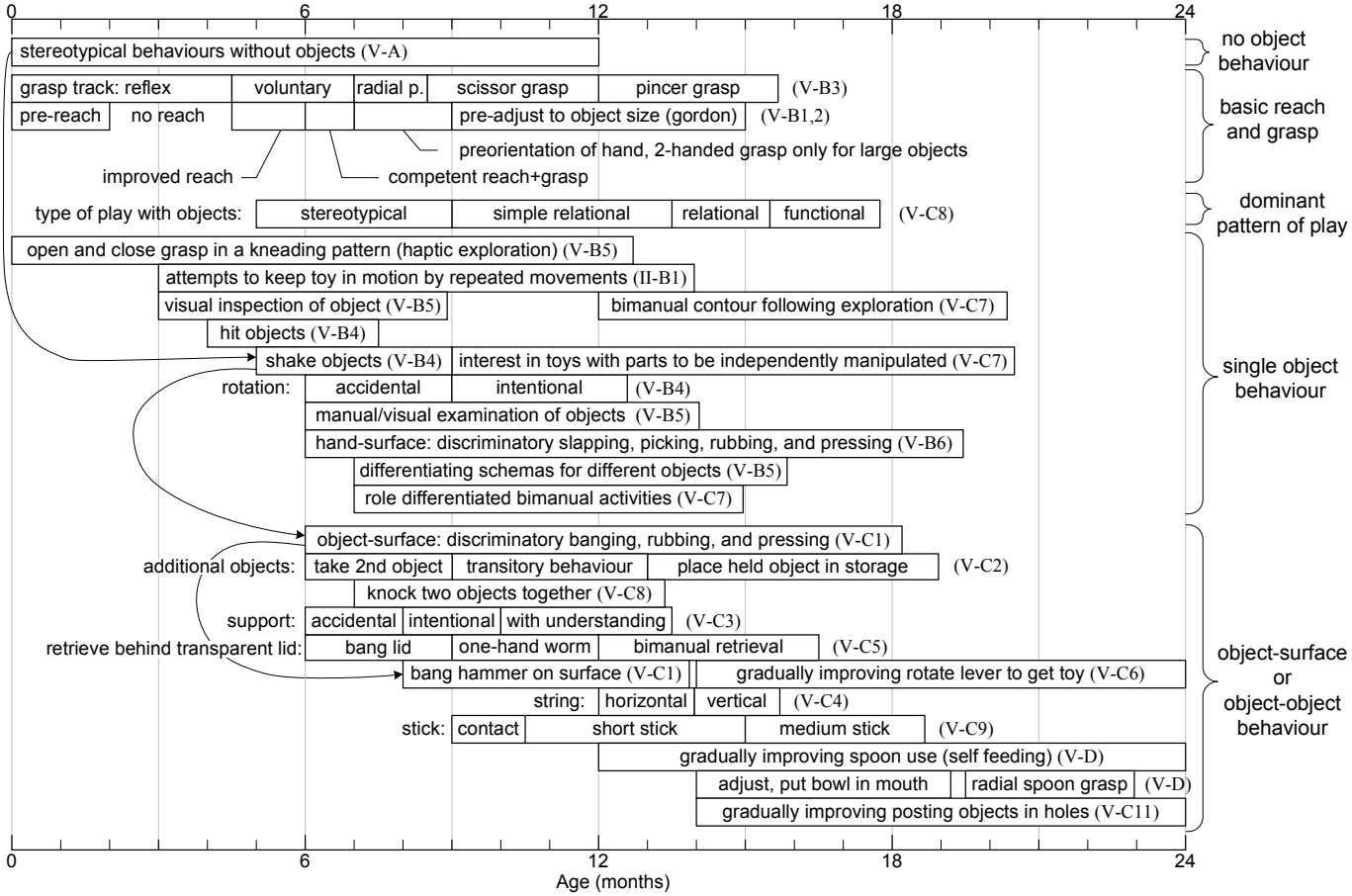


Fig. 2. Behaviours at various ages. The left hand side of each rectangle indicates the age at which the behaviour appears; the right hand-side is meaningless. Cross references to the part of this paper which describes the behaviour are given in parentheses. The arrows show an example of a chain of supposed precursor behaviours. Some of the ages here come from Uzgiris and Hunt [58] with the warning that they "should not be interpreted as typical for infants in general", they are provided to indicate an order of progression, rather than precise timings.

Other stereotypical behaviours relevant to tool use are only performed with objects, and so are covered in Sec. V-C1.

The percentage of the infants' time spent engaging in these movements rose during the period from about 1 month of age through to 6-7 months [18], after which it plateaued and then fell off towards the end of the first year. The average time spent engaging in the movements at the peak was approximately 9%. Although the overall frequency of stereotyped behaviours declined in the second half of the year, the number of different types of behaviours rose, because new behaviours were added without the loss of older ones (unfortunately the study did not report on which specific behaviours appeared at which times). Behaviours tended to be more variable around the time of their first appearance than later (which may be the mechanism of *variation* followed by *selection*, (M2)).

Since the behaviours described above have no obvious precursors, and given that older children and adults do not perform these behaviours, it is not possible that they were imitated, and it is surmised that they are innate. Furthermore Thelen states that "the onset of particular stereotypies is largely dependent upon events intrinsic to the infant" [60]. Thelen suggests that the behaviours are not much affected by the environment, but rather "internally guided" [60]. Thelen suspects that the behaviours may emerge as by-products of the normal maturation of motor control circuits, but that they may be opportunistically used by infants for the purpose of bootstrapping further development, for example by encouraging actions which will at some point lead to interesting results. Evidence that spontaneous behaviour such as kicking can be transformed into an instrumental behaviour comes from Rovee and Rovee [61] (as early as ten weeks) and also Piaget's work (his second sensorimotor stage, approx. 4 months). The general feature of this is accidental discovery followed by intentional exploitation, and is a theme we have already seen in Sec. II-B (for means-ends behaviour) and which we will see arise in many later behaviours. Furthermore Thelen [18] speculates that these stereotyped movements may be incorporated in hierarchically structured advanced skills (*composition*, M5), where the stereotyped movements form low-level subunits.

Apart from stereotypical movements, calibration of the vision, motor, and proprioceptive systems seems to also take place in the first months. Infants may regard their hands moving in the second month, but the vision does not guide the hands [2, p. 102]. Subsequent to this, vision augments the activity of the hand. Infants engage in extensive self-exploration by 2-3 months [62]. Young infants, when viewing their own movements, are sensitive to visual-proprioceptive contingency; infants of 3 months are able to discriminate between direct

and delayed views of self-produced leg movements[13]. Rochat [62] concluded that there is evidence for a perceptual based *body schema* at this time. It would be interesting to know how this body schema can predict interaction with seen objects. Bower [63, p. 123] has shown that there seems to be an innate knowledge[14] of when a primitive reach should contact a seen object because infants were distressed by a "virtual object" which they could see and reach for, but which produced no tactile sensation. However this expectation of contact is likely to be very specific to reaching for a target; we are not aware of any experiments which determine when the infant displays knowledge of expected collisions with objects which are not the targets of reaching.

In summary these behaviours without objects give some of the initial sensorimotor schemas which form the beginning of the developmental story illustrated in Fig. 1; together with the basic mechanisms M1–M6 they start the developmental process, and become *differentiated* (M3) and *composed* (M5) to produce new schemas. This happens because these initial behavioural patterns cause interesting events to occur (touch, sound etc.), leading to *differentiation* of schemas (M3); subsequent *variation and selection* (M2) helps to fine–tune these new schemas, so that they become increasingly predictable (and potentially intentional). Additionally, some of these innate patterns seem to trigger multi-sensory expectations indicating an innate multi-sensory experience space [64]. These behavioural patterns play an important role in learning a body schema.

B. Behaviours With Single Objects

1) Learning to Reach: Reaching is an obvious precursor to dealing with objects, but also it is probably particularly relevant to simple tool use because similar problems seem to be involved: For example in learning to control a stick there is a similarity between learning to bring the hand in contact with a seen object, and learning to bring the stick (which could be viewed as an extension of the hand, see, e.g., [65]) in contact with an object. One needs to solve a degrees of freedom problem to control joints, and a scaling problem to apply the correct force to each element; furthermore there may be visual feedback (servoing) to control the movement to the target. It is probable that evolution has developed innate routines to bootstrap the development of reaching, and it is plausible that some of that innate machinery may be reused for tool use.

Neonates seem to have a very premature reach and grasp mechanism which reduces in frequency over the first two weeks, and is hard to elicit in the period from four to 20 weeks [63], [66, Ch. 6]. The neonate primitive reach motion is visually elicited, and ballistic, with no visual feedback to correct the reach motion while it is in progress [31, p. 38]; it has a relatively low success rate (9-40 percent [19, p. 250],[63]). Bruner [67] suggests that this type of innate response is coded in by evolution to serve as a "launching stock" from which skilled actions can then be constructed. The mechanism of *variation and selection* (M2) seems to be important in building on such innate patterns to develop mature reaching, as shown by Thelen et al.'s [68] detailed study. One striking result of their study was that there were dramatic differences in how four different infants first reached for toys, and how they developed; each seemed to have their own developmental pathway[15]. There is little evidence that the capability to grasp accurately is present in the first month, although some elements such as hand opening are present [31]. The hand opening disappears at about two to three months, and the behaviour becomes more of a swiping with a closed fist, which is then replaced by an open-handed reaching, but with no grasp [69]. The reduction in reaching at about 7 weeks seems not due to any loss of interest in near objects, because visual fixation increases; it is possible that the reduction is because of the excitement of looking inhibiting the motor response, but also there must be some reorganisation of the motor control to explain how the hand opening becomes separated from reaching [70].

It is not until about the fourth month that proper visually elicited grasping will commence, and that the infant will bring the hand into view to grasp seen objects even when the hand is not initially in view [58, p. 110]. The more mature reach and grasp which appears at about five months has about an 80 percent chance of contacting the target, with the possibility of visual feedback to correct the reach in progress, although the grasp coordination appears to have regressed [63]. It seems that the primitive reach-grasp was undifferentiated (i.e., the reach and grasp are coupled), and by 20 weeks two separate motions have replaced it (*decomposition*, M6), which can now be executed independently, or coordinated. Gordon [71] also notes that this type of regression followed by advancement is common: reorganisations can initially result in a decline of the motor skills, before improvement. After this period the coordinated reach and grasp develops rapidly; Bower reports 100 percent accuracy on visually elicited reaching [63, p. 174] at six months.

Studies with prisms which distort the infant's vision show that the infants can use the visual feedback to correct their reaches [72]. However, the visual feedback does not seem to be necessary for normal successful reaching; studies of reaching for glowing objects in the dark [72] conclusively show that sight of the infant's limb is not necessary either at the onset of successful reaching or in succeeding weeks; proprioceptive information must therefore be employed. Nevertheless, vision does seem to be important in normal infants' development of a common mapping system for auditory, proprioceptive, and visual information, because reaching in blind infants is substantially delayed [72].

2) Refining the Reach and Grasp movement parameters: Bruner [67] sees a commonality between the development of reaching to grasp, and the development of other goal-directed skilled actions. In each case the behaviour starts out with a

[13]Leg movements are also included in Thelen's stereotypical behaviours, but we have omitted them above because they seem less relevant to tool use.

[14]His youngest experimental subject was 4 days old

[15]Such development pathways give useful material which helps in the discovery of the underlying mechanisms of development. Also, they illustrate a degree of robustness and generality in these mechanisms, in that they can cope with a variety of different configurations.

series of component acts (for reaching these include raising of arms, ballistic flinging, and closing hand), but the sequence is not correct and the acts are crude; once the sequence becomes correct each component is "drastically altered to fit task requirements". With more practice the whole sequence becomes energy efficient, which suggests that there may be a feedback in the system based on efficiency. Eventually the whole sequence becomes *modularised* (M6), so that it can appear as a component in new higher order sequences.

There is a developmental progression in the infant's use of information about the size of objects, for grasping; infants as young as eight weeks make more reaches to a graspable ball than one that is too large [31, p. 43]); 5-month-olds tend to reach with two hands regardless of size, whereas 7 to 8-month-olds use two hands for large objects more often than for small ones, and at 11 to 12 months reaching closely reflects the object's diameter [73]. A similar pattern appears for the thumb-index finger angle opening during the reach, which increases after 7 to 8 months, as well as the adjustment of the angle to the object diameter, and the proportion of the object within hand opening at touch [73]. In studies of 5, 9 and 13-month-old infants [31, p. 44] it was found that all infants began hand closure before contact, but younger infants began closure later.

For orientation, it was found that 9-month-olds rotated their hand to adjust to the orientation of a stick before grasping, whereas 5-month-olds did most of the adjustment after contact. A detailed longitudinal study [69] has shown a qualitative change in grasp preorientation occurring between 5 and 7 months, which is roughly inline with other results [74]. Investigations of the importance of vision to preorientation [75] have shown that 7-month-olds could preorient correctly to grasp a glowing rod in the dark, showing that visual monitoring of the hand's orientation was not necessary. For 9-month-olds it was shown that they could orient correctly and grasp the rod if shown the rod only before reaching, but the rod remained darkened during reach onset and grasp. Again (as in the previous section) proprioception must be used here, and the authors suggest that vision is used to calibrate the proprioceptive information from the limbs, and thereafter vision is not necessary.

These results on size and orientation show that fragments of a practical object representation are present at this early stage (see also Sec. VI); by "practical" we mean that it may be encoded via the motor action which can grasp it. Such representations allow for the distinction of object size and orientation, while the strategies to adapt grasping appropriately evolve over time. We surmise that these early fragments of representation form the basis for later more complete and action-independent representations (see Fig. 1 upper track). FG: can we link to related robotics work?

3) Development of grasping competences: Infant grasping abilities develop throughout the first year. There is an initial reflex "neonatal palmar grasp" present from birth, where the fingers close on stimulation of the palm; there are conflicting results [76] about whether this disappears gradually over the first year, or disappears after two months, or gradually merges into voluntary grasping.

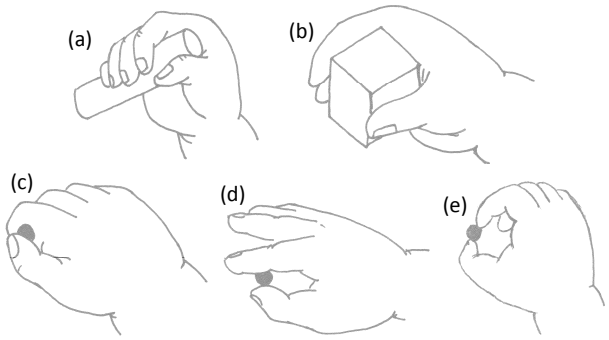


Fig. 3. Developing grasps: (a) palmar grasp; (b) radial palmar grasp; (c) scissors grasp; (d) inferior pincer grasp; (e) pincer grasp.

For voluntary grasping, Touwen [76] in his longitudinal observations of 27 male infants noted a development through five different phases of grasping: the *voluntary palmar grasp* (Fig. 3 (a)) uses the whole hand (appearing on average at 4 $\frac{1}{2}$ months); the *radial palmar grasp* (Fig. 3 (b)) is performed mainly with the area of the palm between the thumb and the second and third finger (i.e. with thumb opposed) (average 7 months); the *scissor grasp* uses the volar sides (volar = the lower surface of the hand, i.e. which includes the palm) of the extended thumb and index finger (average 8 $\frac{1}{2}$ months), or alternatively the side of the curled index finger [77] as shown in Fig. 3 (c); there follow some intermediate stages before the proper pincer grasp: the *scissor-pincer grasp* uses the tip of the index finger and the volar side of the extended thumb (average 11 months), an alternative *inferior pincer* uses the tip of the thumb and volar surface of index finger ([77], Fig. 3 (d)); the *pincer grasp* (Fig. 3 (e)) uses the volar surfaces of the tips of the thumb and index finger almost exclusively (average 12 months). Development is by no means finished here; the second year will see an improvement in the use of appropriate forces, and the grasp will not approximate adult performance until 6 to 8 years, with further subtle improvements in adjustments of force and anticipatory control continuing until adolescence [71]. This gives potentially valuable indications for roboticists how the control of artificial dexterous hands can be learned in some hierarchical scheme starting from a coarse to a fine grained representation. FG: can we link to related robotics work?

4) Developing the Stereotypical Behaviours with Objects: Once reaching and grasping are mastered (c. 5 months) the stereotypical movements of Sec. V-A (without objects) can easily be performed when an object is grabbed, and this is exactly what happens (due to the mechanism of *repetition* (M1)). Thus arm waving becomes waving with an object held in the hand, such as a rattle. The opening and closing of the fingers can be done with an object, either to catch (and release) it or scratch it. These behaviours lead to new interesting effects, and so are reinforced (corresponding to Piaget's third stage which occurs from about 4 to 8 months). This is quite a clear example of the process of *differentiation* of schemas (M3), which can be followed by *variation and selection* (M2) to refine the newly differentiated schema. Other behaviours appearing include rubbing an object with the hand, or squeezing an object, or crushing (e.g. paper).

Some developments with a superficially similar appearance to tool use can also appear at this time, for example an infant can pull strings to shake something tied to the end, or if the infant is in possession of a stick and it accidentally hits an object, then this action can be repeated (this "accidental to intentional" pattern is a feature of the *repetition* mechanism M1). However the combination is discovered by accident and the relationship between the objects is not understood; this is shown by the fact that strings which are close, but not connected to the object, will also be pulled [2], and in the stick example, there is no ability to control the direction in which the stick is pointing.

Infants may also rotate an object during random exploration at 6 months [2], although it is doubtful that it is done with the intention of seeing the other side, as they are incapable of fully turning objects to find a hidden desired side until about 9 months [58, p. 120]. They may however intentionally half-rotate an object to bring a seen desired part to the mouth [2].

5) Multi-sensory Object Exploration: There is an obvious progression in the sophistication of the infant's object manipulation abilities, but in parallel with this there is the less obvious development in the infant's perceptual abilities. It is surmised that in some instances the behavioural developments help the perceptual development [78], [79]; this has been studied in the case of haptic perception: Before reaching to grasp is mastered, objects are manipulated, and information about them is gathered haptically. Infants haptically perceive object size probably during the first months of life [78]. Newborns have been shown to discriminate objects by haptic exploration (e.g. cylinder vs. prism) based on shape [64]. More specifically, infants of 2 months can gather partial knowledge of shape from clues such as points, curves, presence or absence of a hole [64]. We saw in relation to grasping (Sec. V-B3) that a visually-based object representation was forming, now we see here that a haptic based representation is forming in parallel. FG: can we link to related robotics work/haptics?

There is also evidence of cross-sensorial transfer in neonates [64]; i.e., infants who were habituated by haptically exploring an object were subsequently able to visually discriminate between that object and another novel object. This transfer from touch to vision has also been shown to be present at 2 months, but transfer in the other direction is absent (i.e., infants do not haptically discriminate between two objects if they have just seen one of them) [64]. Interestingly, at 5 months (when reaching and grasping are coordinated) the reverse has been shown: infants transfer from vision to touch, but no longer from touch to vision [64]. It has been surmised that the two senses may have their own representations developing at different rates, and at certain times the level of representation in each one might not facilitate transfer; the haptic system seems to be present very early and to mature slowly, whereas the visual system appears later but develops more rapidly [64].

Infants can perceive hardness or compliance by 6 months and possibly earlier; the development of such perception may be facilitated by the main action performed with objects for about the first three months, which is to grasp in a fist and open and close in a kneading pattern. Infants can perceive tactile texture[16] by 6 months, but not earlier [78], [80]; this may be because it relies on practice with exploratory rubbing movements. In normal contexts infants perceive weight probably about 9 months, which notably comes after they have experience with waving objects, although there are exceptions [80]. In darkness, when infants are seated upright on the parent's lap, 3-month-olds can perceive weight; the darkness removes visual stimuli that may be consuming the infants' attention, and the infants' posture means that as soon as they have possession of the objects they have to support their weight (as opposed to fingering them on a table) [81]. There is also some evidence that properties of temperature, texture, and compliance may be perceived by 3 months if in the dark [81]. These advanced results in darkness highlight the difficulty of determining the competences of infants; failure to perform a task might not reflect a lack of ability, but rather it may be simply because a competing stimulus was more exciting.

The above examples show how behaviour may facilitate a perception. In the other direction infants' behaviours with objects are affected by their haptic perception. Lockman presented infants at ages 6, 8, and 10 months with hard and soft objects [82]. Infants at all ages squeezed the soft object significantly more than the hard object, but squeezing of the soft object increased significantly with age.

To model similar behaviours in artificial systems, sensors comparable to the human system are required. While powerful visual sensors are readily available nowadays the selection of available complex tactile sensors is very limited. These sensors show a significantly worse performance compared to human abilities in some or all of the following dimensions: spatial resolution, sensitivity, disparity between capabilities of hands and capabilities of sensors, long term stability and system integration [83]. In this sense, further progress on tactile sensor development needs to me made before similar behaviours can be replicated in artificial systems.

6) Hand to Surface Interactions: Hand to surface interactions tend to occur after manipulation of objects because the infant usually needs to be seated (with some assistance, about 6 months [78]) in order to access surfaces. Interactions between the hand and a surface can be considered to be quite similar to the hand with an object, and again the rhythmic flexion and extension of fingers (described above [60]) was also often performed to scratch a surface. Lockman specifically studied surface interactions; he presented infants at ages 6, 8, and 10 months with surfaces which were liquid, discontinuous (net), flexible (sponge), or rigid [82]. He recorded actions of slapping, picking, rubbing, and pressing; these actions may themselves be derived from stereotypical behaviours (e.g. slapping from waving) or recently acquired grasping actions (e.g. picking).

He found that infants discriminate, for example they pressed a flexible surface more than the other three, and rubbing was more prevalent across liquid; furthermore the discrimination develops with age, becoming more pronounced. Overall then

[16]Texture here means the fine grained property distinguishing same-shaped blocks if they were covered with, e.g., a towel, rubber, smooth cloth, or plain wood. Two objects appearing the same to the eye may have different textures, so the term is not synonymous with its use in computer vision.

we see that once infants are grasping and acting on objects (or surfaces), they begin to discriminate the properties of those objects, and this must link to developing object representations, which are beginning to be formed. By now the infant understands something of the properties of individual objects (as a result of *differentiation*, M3); this means that the sensorimotor schemas (e.g. for banging, pressing) include sensory abstractions which discriminate between different objects and surfaces (in order to predict different consequences for the action being executed on each one). This discriminating knowledge forms a substrate which will allow the infant to progress to learning about the effects of actions involving relationships among these objects and surfaces.

C. Object-Object Behaviours

This section covers the early object-object interactions which involve controlling relationships among objects, and through which knowledge is acquired about relationships between objects.

1) Object to Surface Interactions: Of Thelen's stereotypical movements [60], two were only performed as an object-surface interaction. One movement consisted of an infant holding an object and rubbing it (horizontally) against the surface of a table or floor, with movement from the shoulder. The second was a push-pull movement from the elbow (flex and extend) with the arm parallel to the floor or table. This was typically done for an object too heavy to be lifted, so instead it was pushed back and over on the surface. Thus it seems that this arises due to the interaction between an innate motor behaviour (push-pull of elbow) and constraints of the physical world.

The stereotypical waving action has been performed with objects, as noted in Sec. V-B6; now when a hard surface is present this can lead to the behaviour of banging on the surface and producing an interesting sound. Lockman has studied how this banging action becomes discriminatory (a result of *differentiation*, M3). Again, he presented infants at ages 6, 8, and 10 months with surfaces which were liquid, discontinuous (net), flexible (sponge), or rigid, and also with hard and soft blocks [82]. In this case there was an inability of the younger infants to discriminate some of the relationships. All infants banged differentially, banging more frequently with the hard block on the net and rigid surfaces, but only the 10-month-olds banged the hard block more than the soft one. Also only the 10-month-olds differentially rubbed across surfaces, rubbing over the rigid one for longer. This again suggests that infants develop by first focusing on the exploration of individual objects and only at a later stage on object relations. As a consequence by 10 months there is a general improvement in the ability to handle relationships between a grasped object and a surface it acts on. Possibly the limitation of younger infants is important because it prolongs the period of dealing with simple relationships, so that they can be learnt thoroughly.

A further study [84, Ch. 21] tested infants monthly from 8 to 10 months with hammers to bang surfaces. The hammer heads were hard, soft, or half hard/half soft. The surfaces were hard or soft. Infants at this age were able to hold a hammer by its handle, to use as a tool. It was found that all ages banged the hard hammer more than the soft hammer, on the hard, but not soft surface. Furthermore there were more hits with the hard side of the mixed hammer.

This action then is almost tool use, however, orienting a hammer to a surface is easier than directing it at a specific object. When an infant is presented with two surfaces on a table side by side, and is able to selectively bang on one surface, then the infant shows awareness that these two surfaces are distinct. This is very close to selectively banging against another object, and forms a possible bridge to banging a held object against a stationary one.

Overall the path we have traced shows how the original stereotypical movements could help to bootstrap the development of object-object actions primarily via the mechanism of *differentiation* (M3), which is itself triggered by accidental discovery following *repetition* (M1).

2) Taking an Additional Object: As soon as infants can grasp one object they will inevitably face situations where they want to grab another even though they are already holding one. Bruner [15] examined the way infants respond to being handed multiple toys, one after the other. Infants from five different age groups were tested.

At 4-5 months some infants could not even get the first toy, and some could not hold it for long. Some infants succeeded in taking the second toy, but only because they inadvertently dropped the first before taking the second. In general infants tended not to grab the second toy if the first toy was already in the process of being taken to the mouth.

At 6-8 months good command of the grasp was attained. On being presented with the second toy these infants often transferred the first toy to the other hand to free the preferred hand for reaching. The development of this behaviour came from taking the first object to the midline in order to hold it with the two hands, and then reaching out with the nearest hand; and this then evolved into an anticipatory handover. Sometimes instead of the transfer the infant would reach across with the empty hand.

At 9-11 months one fifth of the trials successfully dealt with 3 to 4 objects. The strategy employed was to put one in reserve storage (in the lap, or beside the infant) to free the hand for the second, although this often (50%) triggers another grasp attempt immediately, i.e., the infant forgets why he put down the first object and/or cannot inhibit the action of retrieving it again immediately (see again Sec. II-B on affordance based to goal based action). To overcome this difficulty requires a capacity to delay the retrieval response, and to maintain the intention for grasping the new object. These abilities are obviously also important for more complex problem solving requiring planning (Sec. II-B).

At 12-14 months the storage strategy was well developed, and furthermore the infants can place an object in storage before a third or fourth are handed to them. At 15-17 months the mean number of objects which the infant can take possession of has gone from 3.0 to 3.7, and objects are stored in one way consistently. Overall from 6-17 months there is a gradual increase in leaving it there, rather than a sudden step change. The development shows a process of integration of the constituent acts into a successful behaviour

(see *modularisation*, M6) [15]. We can also see that knowledge of space and special locations is necessary for dealing with more than one object. In the earlier interactions with a single object the reach and grasp were triggered, and then the infant manipulated the object. Location was implicitly coded in the reach behaviour, but the infant was not forced to be aware of this. However, when two objects are being handled the infant is forced to become aware of locations apart from the location implicit in a reach. We have mentioned the beginnings of object representation before; we see here the beginnings of spatial representation, which becomes more abstract in a similar way (see abstract track of Fig. 1).

3) Pulling the Supporting Object: Willatts analysed the task of pulling a towel to retrieve a supported toy from 6 to 8 months [85]. He recorded not only success or failure on the task, but also monitored the infant's gaze, in order to have an objective measure which could discriminate between accidental success, or intentional success. Younger infants (about six months) tend to give up on the toy and play with the towel instead, but in doing so they often accidentally bring the toy into reach. Willatts was able to monitor the infant's gaze, and to show that there was a transition: whereas the younger infants (6 months) gave up on looking at the toy, as they got older, there were more glances towards the toy (8 months), suggesting that pulls of the towel were intentional in order to retrieve the toy. These kinds of accidental discoveries lead the child to understand the effects of various actions on object-object relationships, and lead to the development of a repertoire of "means" actions which can be employed to achieve goals [2], [20]. It is an example of the mechanism of *repetition* (M1) leading to accidental discovery followed by *composition* (M5) of the means-end behaviour, which then allows intentional exploitation.

Note that in the case of the support the necessary relationship (on top of) is not understood at 8 months, and up until 10 months or later the infant will still pull a support even if the desired object is held above it and not touching it [58, p.111], or resting on an object close to the support [2, p.283].

4) The String: A string is tied to an object and must be pulled in order to bring the object within reach. The string is particularly easy because of its unbreakable contact with the object [86]; it is hard to go wrong (in contrast to the stick, Sec. V-C9), if the string is shaken wildly the object will still not be lost from the end of it, and is quite likely to be brought closer. Uzgiris and Hunt [58] tested two different string situations; the easier situation is on a horizontal surface where retrieval of the desired object can be performed by repeatedly pulling the furthest part of the string towards oneself, and then letting go. The more difficult string behaviour is when the object must be raised vertically; the horizontal strategy fails here because the object falls if the string is released; success requires bimanual control, typically with one hand pulling, and then passing control to the second hand which prevents the object from falling, while the first hand stretches again. Uzgiris and Hunt observed the horizontal string task at 12 months, and the vertical string task at 13 months [58, p. 111].

5) Obstacle Removal/Avoidance: This behaviour is a step towards tool use because the relationship between two objects must be acknowledged (obstacle and desired object), and one must either remove the obstacle or detour around it. Learning the means-end coordination to remove an obstacle in the way of grasping is one of the first means-ends behaviours described by Piaget [2, p.217], which he places at $7\frac{1}{2}$ months; as with the support, it may be learnt by an accidental discovery followed by intentional exploitation. This is a difficult problem for infants at this age because they are not used to dealing with two objects, and so it hard for them to execute an action on an object (obstacle) which is not the current goal; Piaget speculates that other two object behaviours such as placing one object aside in order to take another (Sec. V-C2) may derive from obstacle removal [2, p.217].

Bruner [15] looked at the task of retrieving a toy from behind a transparent lid. The lid could be easily lifted, but fell down if not held open. The behaviours observed in infants were very much in line with Siegler's multiple strategies in overlapping waves (Sec. III-C); infants used various strategies each of which had peak at a certain ages and decreased only gradually. The youngest (6- to 8-month-olds) went directly for the toy and then engaged in banging of the (closed) lid, which may have become an end in itself; this behaviour gradually decreased with age, but still appeared in some trials for the oldest infants. Infants of 9 to 11 months predominantly used two different strategies: (i) raising and closing the lid, which also seemed to become an end in itself; (ii) raising the lid with one hand and carefully "worming" the same hand into the opening so that the hand (and arm) prevent the lid from closing. The fourth strategy was two-handed: the lid is opened with one or two hands, followed by a reach with one or two hands, but the lid is not held open long enough for efficient retrieval. This behaviour had some presence in all groups, gradually increasing and peaking for the oldest (15 to 17 months). The final strategy involved holding the box open with one hand while the other hand retrieved the toy; this increased sharply after 12 months. Thereafter there was no new strategy, but this strategy became less effortful and quicker. Progress to the final strategies depends on advances in bimanual control, and so this is an example of where progress to a node (i.e. behaviour) in Fig. 1 (concrete track) may have to wait for all its necessary precursors to be ready.

6) Rotate a Lever: This task involves a 42 inch lever which can rotate about its centre on a table. One side of the bar is within reach of the child, but the far side is inaccessible. An attractive toy is tied to the far side. The child must rotate the bar in order to bring the toy around to the reachable area. This is difficult because the child must take the unusual action of pushing the bar away in order to bring the toy closer.

Koslowski and Bruner [87] tested children of three age-groups (12-14, 14-16, and 16-24 months) and categorised the strategies they used as follows: (i) Linear: reaching directly for the toy, trying to push the bar in a straight line towards or away, pulling the table. (ii) Oscillation: pushing the bar back and forth, but never rotating more than 45° from the midline, and tending to return it to midline after rotation. (iii) Partial rotation: rotating 45° and then stopping to consider the new position, but not making a concerted effort to reach for the toy. (iv) Absorbed in the rotation activity, often rotating

the toy within reach, but ignoring the toy (see comments on affordance-based play in Sec. II-B). (v) Rotate and capture.

There was a progression towards more advanced strategies with age. Younger children found it difficult to suppress the linear strategy, and this explains the oscillation strategy: after rotating a bit, they resort to pulling the bar straight towards them (hence returning it to midline). Repeated failure with this almost forces the child to consider unidirectional rotation. Thereafter the child can pay attention to two aspects of the apparatus: either the relation between movement of the bar and the position of the goal, or the way in which the movement of the bar can be effected. The authors suspect that both cannot be attended to simultaneously due to information processing capacity limits (Sec. II-C); therefore they must be first *modularised* (M6). While focussing on looking at the toy, little progress is made in unidirectional rotation, on the other hand focusing on the rotation leads to strategy (iv). Eventually the fact that the goal is within reach is noted; this sort of accidental discovery bears some similarity to the discovery of the support (Sec. V-C3). The change in strategies used was inline with Siegler's overlapping waves theory (Sec. III-C); there was a marked increase in the use of strategy (iii) by the 14- to 16-month-olds, and the 16- to 24-month-olds had the largest number of children using strategies (iv) and (v), but older strategies had not died away completely. This is also inline with the idea of schemas being the unit of behaviour: the novice child has a well developed schema for pulling in a straight line, but is only developing the schema for rotation; the child must ignore the goal in order to focus on developing the rotation schema further, so that it can be later used as a means action (see Sec. II-B). The child's modular approach to the problem has a major benefit: "Not only is the problem solved, but it is solved for a wide variety of circumstances and forms in which it is likely to be encountered, wherever the lever may point, whatever its shape, and so forth. Transfer, so to speak, is built into the solution." [87]

7) Advancing Bimanual Control, and Object Manipulation: Bimanual control is required in many tool use scenarios, and it is reckoned to be an important component in explaining why human tool use capabilities exceed those of other animals [84, Ch. 24]. Behaviours such as holding an object in one hand, and striking or stroking it with the other are the beginnings of "role differentiated bimanual activity", and appear as early as 7 months [88]. Some toys are more likely to elicit bimanual activities than others at particular ages, but overall the frequency of bimanual activities increases linearly with age [88]. Infants of 7 months were as likely to execute bimanual activities on toys with no movable parts, as those with moving parts, but from 9 months onwards toys with no movable parts elicited few responses, and infants seemed more interested in toys with parts to be independently manipulated [88]. It is surmised that these developments require a combination of neural developments, as well as having the appropriate objects, and also understanding the properties of those objects [88]. More sophisticated bimanual actions appear towards the end of the first year. For example "contour-following" can be observed at 12 months, which involves holding the toy in one hand and manoeuvering it, while the fingertips of the other hand are moved smoothly over its edges [89], or employing a single finger or pincer action for manipulation with the second hand at 11 or 13 months [88].

8) Relational play: Apart from intentional problem solving, there is also a natural progression towards object-object interactions in infants' free play. An overview of infant free play is shown Fig. 2 (just below the grasping track). When infants of various ages were presented with a wide variety of toys, three categories of play were observed [90]: stereotypical play was dealing with a single object (mouthing, fingering, waving, banging) and dominated at $9\frac{1}{2}$ months; relational play dealt with associations of two or more objects and dominated at $13\frac{1}{2}$ months; functional play was using a toy in a manner deemed appropriate to an adult, such as using a comb to comb a doll's hair; it dominated at $15\frac{1}{2}$ months. This study shows that by $13\frac{1}{2}$ months most infants prefer to explore relationships among objects, rather than exploring objects individually. A further study [91] gave more insight into the precursors to full relational play; at 7 months the very simple relational action of banging two objects together was common, and by 9 months infants could do very simple relational acts such as touching a spoon to the base of a pot; it was between 9 and 13 months that most infants made the transition from these simple relational acts to "accommodative" relational acts such as putting a lid on on a pot or a spoon in a cup. It is around 10-11 months that infants begin to establish the links between particular objects and their "canonical actions", e.g. a hammer is for banging, a brush is for sweeping, and also the spatial relationships that must be established between tool and target object; e.g. the relations "in" (key or screwdriver in a slot), on (one block on another), under (put a spatula under a pancake) [80].

In conclusion there is a significant advancement in the infant's knowledge of spatial relationships during this period. It begins as a very practical knowledge of a relationship between two concrete objects discovered during play, but it is quickly generalised to other similar objects, and thus becomes a more abstract knowledge (increasing abstraction is shown by moving to the right on Fig. 1, abstract track).

9) The Stick: The behaviour of the stick entails using a stick to move an out-of-reach object and bring it within reach of the hand There is in fact a spectrum of behaviours under the broad umbrella of "the stick". The simpler end of the spectrum consists of using a short stick to retrieve a barely-out-of-reach object with a single sweep of the arm; the arm is initially extended so the stick reaches beyond the object, and then it is brought towards the body (which may happen towards the end of the first year). The more complex end of the spectrum includes behaviours using a long object (e.g. a long stick or mop) to knock an object from side to side until it can be reached (which may be placed at about 3 years [92]). Uzgiris and Hunt tested a medium-length stick (18 inches long) [58, p. 150]; They place the behaviour at 15-18 months [58, p. 111]. It is a relatively difficult behaviour when compared to strings and supports because the tool is not given in the appropriate relationship from the outset; the infant must create the appropriate relationship.

Brown [92] tested children ranging from 17 to 36 months old for transfer ability in retrieval tasks with a variety of stick-

like tools, some of which had a hook or rake at the head. Tools varied in length, rigidity, colour pattern, and type of head. Children never selected non-rigid tools. Overall the children seemed to understand quite well the properties required of an effective tool for the task. Brown makes a strong case for the ability to transfer being very much domain specific, and related to the child's understanding of causality in the particular task.

10) Perceptual aspects in retrieval tasks: Bates et al. [86] looked at perceptual aspects in retrieval tasks for 10-month-olds using support, string, stick, and also a hoop and crook (stick with semi-circular hooked end). It was found that if the tool and desired object both had the same colour and texture, then it was particularly difficult for the infant to succeed. It was surmised that the perceptual difference may help the infant to discriminate the two objects and to keep them both in mind as separate entities. A difference in both colour and texture was no more helpful than a difference in one or the other.

The effect of the spatial configuration of the objects as presented to the infant was also investigated. Four types of spatial configuration were presented: unbreakable contact (support and string), breakable contact (hoop, or crook, presented in contact so that the tool only needs to be pulled), behind (hoop, or crook, surrounding object, but behind and not in contact so that the tool only needs to be pulled), beside (hoop, or crook, or stick, presented beside each other, so that the tool needs to be brought into contact before it is pulled). It was found that difficulty increased as follows: unbreakable contact, breakable contact, behind, beside. The four tasks in the "breakable contact" and "behind" groups all required the same motor action (pull the tool), yet there was a significant difference in success on them with the hoop in contact being significantly easier than the crook behind. This suggests that the infant understands the causal relation when two objects are connected and the physical contact may help the infant to remember this. It is not likely that the infant conceives of the connected objects as a single entity, because perceptual similarity of the objects is a hindrance.

11) Fitting Shapes into Slots (peg-in-hole task): The task of inserting a cylindrical peg in a cylindrical slot can be done by almost 50% of infants at 12 months, but they do not preorient the cylinder for insertion [93]. Instead, they press one end to the hole, and then move around the other end until they find the right orientation. By 16 months infants do preorient the cylinder, but not other shapes, until later (note: context specific skill, see Secs. III-C, III-D). The 12-month-olds seem to use the reduction in degrees of freedom (DOF) strategy described in Sec. II-A: they first hold the peg in a fixed orientation in the hand, and move the far end of it into contact with the hole. This is a three DOF problem (which is very similar to controlling the hand in a reach towards an object). The second step is to orient the peg to be parallel to the slot, while pressing it into the hole so that the end in contact with the hole maintains contact. This is a two DOF problem. The cylinder itself has five degrees of freedom, but the sequential approach reduces the problem space. Repeated practice helps the infant to learn the correct orientation, and so the infant tends to approach the hole with a gradually better preorientation in successive trials.

The cylinder is relatively easy as it can be inserted with its cross section in any orientation. Shapes of non-circular cross-section (e.g. triangular or rectangular) must additionally be oriented so their cross section matches the opening. Children are remarkably bad at this task until about 26 months [93]. This seems to reveal something about the object representations they are using (see Sec. VI). In addition, insertion of disks in slots shows that 18-month-olds fail to preorient, even though they can well preorient their hand for insertion in a slot [94], showing context specificity of representations.

12) Objects With Handles: McCarty et al. [95] studied how infants deal with an object with a handle, and in particular what way they grasp it. These experiments were done with a spoon pre-loaded with food and toys with a handle (bell, rattle, cow, pig). Each object has a handle and a *goal-end* (e.g. the goal-end is the bowl of the spoon, or the toy). Three different grasps were categorised as shown in the following sketch:

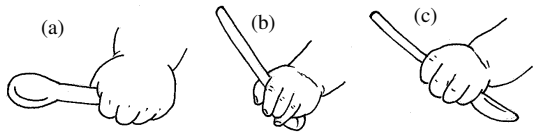


Fig. 4. (a) Grasp the handle with the thumb towards the goal-end of the object (hereafter called *radial* grip); (b) grasp the goal-end itself (*goal-end* grip); (c) grasp the handle with thumb towards the non-goal-end (*ulnar* grip).

This study looked at how infants develop the ability to use the radial grip (Fig. 4(a)) at the start of the task. The preferred hand of the infants was identified in a pre-test. In the results 67% (for toys) to 70% (for spoons) of grasps used the preferred hand. The objects were presented to 9, 14, and 19-month-old infants, on a stand, with the handle alternately oriented to the left and right; trials could then be categorised as *easy* if the object was presented in an orientation which would allow an overhand grasp with the preferred hand to achieve a radial grip (otherwise it was *difficult*; i.e. an overhand grasp with the preferred hand would achieve an ulnar grip). The most interesting results then concern how the infant dealt with the difficult case: 9-month-olds tended to use any of the three grasps indiscriminately; when they used a non-radial grip, they often (more than half of trials) put the handle in their mouth, and typically corrected afterwards. 14-month-olds were less likely to grasp the goal end and more likely to use an ulnar grip; however, when they used a non-radial grip they never put the handle in the mouth; instead they corrected either by rotating the wrist awkwardly, or changing to the other hand. 19-month-olds used the radial grip on 86% of difficult trials, which meant that they had to suppress the tendency to use their preferred hand and use the other instead (see also [96] for the training experience which accelerates this).

Following from these results the authors formulated a model of the development of planning in this task: (i) Feedback-based strategy: after an indiscriminate grasp the end of the spoon near the thumb is brought to the mouth first, and if this turns out to be the wrong end, then a correction is made and the other end put in the mouth; (ii) Partially planned strategy: as soon as the spoon has been grasped its orientation is noted, and if it is incorrect an adjustment is made before bringing it to the mouth (this entails inhibiting the preference to bring it straight

to the mouth); (iii) Fully planned strategy: the orientation of the spoon is noted before grasping and a grasp which is appropriate to the goal of feeding is selected. This model predicts that infants' actions should be slower when planning is taking place; some evidence for this was found in that the action of bringing the spoon to the mouth, when the ulnar grip had been used, was slower in 14-month-olds than in 9-month-olds [95]. Reflecting on development in this task we see a striking lack of planning at the earlier ages, which is inline with the idea that behaviour is more affordance-based before the second year (see Sec. II-B); i.e. an affordance-to-grasp suggests itself and is immediately acted on without regard for later steps.

Further work has shown that the radial grasp generalises to other tools with self-directed goals (e.g. hairbrush on self), but not to other-directed goals (spoon to feed a toy lion, hairbrush to brush toy, hammer to object) [97]. This reinforces the ideas about the context specificity of knowledge (Secs. III-C,III-D). We suspect that developments on the abstract track (Fig. 1) are necessary before the common deep structure in such tasks is obvious to the child (see Sec. VI).

D. Tool Use Example: Transport Using a Spoon and Bowl

Many of the behaviours described so far have been building up the necessary knowledge for tool use by understanding the properties of individual objects (Sec. V-B), and the effects of various actions on various objects in relationships (Sec. V-C). Self-feeding from a bowl using a spoon is "proper" tool use, and is common in human cultures. Connolly and Dalgleish [98] studied two groups of infants longitudinally, at monthly intervals; one from 12-16 months, and the other from 18-23 months. They outlined four stages in the development of this behaviour: (i) repeating one part of the feeding sequence, such as putting the spoon into and out of the bowl, or into and out of the mouth; (ii) performing the outline of the correct action sequence spoon-to-dish-to-mouth, but not effectively loading food on the spoon or unloading in the mouth; (iii) effective performance of loading and unloading within the sequence; (iv) incorporation of correction routines, e.g.: check if food has been successfully loaded; if not, return to the bowl; or pick up food that has dropped during the transfer to mouth.

The behaviour of stage (i) can be called *play*, where the goal of feeding was not pursued, and the means is done for its own sake (mechanism of *repetition*, M1); in addition, the infant would sometimes pass the spoon from hand to hand, bang it in the dish, or on the table, or drop it to the floor, or rub it against his/her own head (mechanism of *variation and selection*, M2). Though these activities were not directly in the service of feeding, they did serve to increase the infant's knowledge of these actions, and their effects in the feeding context (this is the role of play, as described in sec. II-B). Sometimes goal directed behaviour was observed, but there was a lack of understanding of the purpose of the spoon: the younger children were sometimes observed putting their spoon into and out of the dish repeatedly, while taking food from the dish with their other hand. In the behaviour of stage (ii) the younger infants did not seem to understand the need to load the spoon. The behaviour is learned by imitation (Sec. II-D), so they have some knowledge of the sequence of the operations before understanding their individual purposes. To effectively learn a component part (iii) is an example of means-end behaviour which tends to follow the pattern of accidental success leading to acquisition of the appropriate schema (where M1 leads to M3), followed by intentional repetition with variation and refinement (M1 and M2), and later understanding (see Sec. II-B3). The correction of errors in the sequence occurs first for those elements at the end of the sequence, and latest for those at the beginning of the sequence; e.g., by 18 months the remaining errors are only in the earlier stages [99]; this is probably due to the younger infants' limits in general planning abilities as noted in Sec. II-C.

In terms of the component skills, the authors outlined four principal problems for the infant: controlling the spoon in the hand, loading food on the spoon, taking it to the mouth without losing the food, and unloading in the mouth. A number of "behaviour categories" were devised to code the observations of the infants in various activities such as grasp employed, trajectory to mouth, loading method, etc.; e.g., for the loading of the spoon these included: (i) dipping-in motion (spoon lifted and lowered, sometimes repeatedly); (ii) side-to-side motion across the dish; (iii) scooping motion towards the infant; (iv) dropping the spoon in the bowl and sometimes picking it up. Overall the results showed that younger infants had more varieties of hand grasps, and less stable movement strategies; with age came increasing consistency in the actions used (mechanism of *variation and selection*, M2). Also the behaviour categories used changed; e.g., for loading the spoon, younger infants preferred dipping-in, while older ones preferred scooping with a wrist rotation, or the side-to-side motion (which was often effective in trapping food against the side of the bowl) (see Siegler's changing strategies, Sec. III-C). The overall pattern of movements became smoother and more direct, and the time needed to perform individual components of the action decreased (see *modularisation*, M6). In terms of hand grasps, older infants used fewer inappropriate grasps (e.g. ulnar grasps are inappropriate because the arm gets in the way when trying to bring the food to the mouth), and furthermore older infants used more flexible grasps; flexible here means that the spoon can be manipulated with finger movements, as opposed to a rigid grip which only permits wrist movement (these general features are also seen in the progression from novice to expert in adults [84, Ch.4,5]).

We also note a strong similarity with the task of learning to drink from a cup [15, p.72]. Initially the child grabs the cup and pulls it to the mouth in a single step. With practice the child slows this down and puts in a number of stopping points to rebalance the cup so the liquid doesn't spill, and also adjusts the head position, bringing head to cup, and monitoring how the cup is moving towards mouth (see *decomposition*, M6). With more practice it becomes a smooth motion where monitoring of the level is done continuously during the motion (see *modularisation*, M6).

In this behaviour we see how the mechanisms of schema development need to work together over a relatively long time to eventually produce efficient spoonfeeding skill. We should

also point out that this task is relatively simple because it does not require mental representation of unseen parts, which poses more severe difficulty for infants (see e.g. [100]).

VI. Internal Representations (Abstract Track)

This section briefly looks at changes in internal representations (upper track, Fig. 1), using observable ability to transfer as a way to deduce what representations may be in use. Transfer of specific skills to similar, related scenarios or objects is very important for robust tool use. The evidence from Sec. V suggests that improvements in this ability during development can be explained by increases in the world knowledge within the system, rather than some generic developing "transfer ability". We have seen from Brown's study on retrieval [92] (Sec. V-C9) that children transfer very well when they understand the causal relationships in the particular task; Brown has also shown that they do not transfer on more abstract tasks where the relationships are not understood according to any of their prior knowledge and therefore seem arbitrary to them. This message is reinforced by a further study of 3- to 5-year-olds [101] which points out that the ability to transfer is not directly dependent on age, instead it is dependent on the level of representation achieved; young children can achieve a deep representation of causal relationships in tasks involving simple physical manipulations, and therefore can easily transfer in these. Older children can achieve a deeper representation in a wider variety of domains, and hence can show transfer in more domains. Therefore the observable ability to transfer could serve as a proxy for deducing something about the unobservable internal representations. Using this we could say that representations seem to develop in (at least) the following three ways.

1) Coarse to Fine: In some situations infants generalise very well, and immediately (for example supports or sticks [2, Obs. 152,160]). The fact that these generalisations can happen immediately after the skill is first learned suggests that the objects were already represented in the same way (e.g. a coarse representation of a long object); once the skill is learned for one, it can generalise to all. Sometimes infants over-generalise, e.g., *scale errors* [102], where behaviour is generalised to objects of incompatible sizes (such as a too large peg in a hole), or the attempt to insert incompatible shapes in holes; this again suggests a coarse representation, which might ignore some details of shape and scale. The development of representations seems to follow a path from coarse to fine, with initial representations capturing rough shapes, and the detail on objects only being gradually elaborated later.

2) Context Specific to General: In some situations infants do not generalise well at all, for example in the way a spoon is grasped for self-feeding, or for directing to another object (Sec. V-C12), or placing the hand in a slot vs. posting a disk in a slot (Sec. V-C11). Much of an infant's learning is quite task-specific. Examples of lack of generalisation suggest that high level representation is not that well-developed (i.e. the high-level similarity between tasks is not apparent to the infant) and suggest that it is important to spend an extended period focusing on task-specific learning. This then needs to be followed by some process of representational redescription (M7) which can find a higher level abstraction which is common to a number of concrete behaviours. This higher level may for example capture causal understanding of the behaviour, and when it is achieved generalisation in other domains becomes possible, and understanding of demonstrated actions becomes possible as well.

3) Integration of Fragmentary Representations: Kellman and Arterberry explain that "perception leads to multiple representations that may be recruited for different tasks" [19, p. 262]. Part of the work of development is to connect these up to produce more generic and reliable world models. We have seen examples of this already in the connection between haptic object representation and visual object representation (Sec. V-B5). Additionally, Kaufman et al. [103] describe how the two separate visual processing streams in the infant brain (dorsal or ventral) are responsible for different tasks. The dorsal route seems to be primarily used for knowledge relating to grasping (a practical representation), while the ventral is for representation and recognition of the whole object; yet these must be integrated to allow grasp knowledge to be associated with an object representation. It may be at quite a late age (maybe nine months [103]) that infants can integrate the information from the two streams. Both before and after this there is further evidence of integrating fragments. Surprisingly advanced perceptual competences are shown by 4-month-olds in perceiving the 3D form of rotating wireframes [19, p. 168], yet this seems to constitute only a fragmentary understanding of objects because they do not "complete" solid 3D objects until 6 months [104]. Even at 18 months, fragmentary representations based on view dependent images and parts of objects seem to be still in use, and then there is a period of rapid change where 3D whole-object geometric representations are built by 24 months [105]. The picture emerging from the literature suggests that object representations may undergo a long and complex developmental trajectory. At the same time, we can see advantages of fragmentary task specific representations: they provide a simple space which is appropriate to a particular task, and when another seems more appropriate it is possible to switch representation (see also [106]).

VII. Reflection and Recommendation

In this section we firstly reflect on the psychological results to summarise the salient points about how the overall development works (Sec. VII-A); we then formulate some succinct guidelines for developmental roboticists who wish to model similar developmental trajectories (Sec. VII-B).

A. Reflection on Infant Development

In reflecting on the examples above we can see the two tracks of sensorimotor skill and representation developing (Fig. 1). From this we extract the following main ideas:

1) Innate Knowledge is Fragmentary and Incomplete: Innate knowledge of the physical world seems to be given in a fragmentary form; it is not given from the outset in the useful form which an adult has, but rather the evolutionary endowment seems to provide constraints and boosts for the

development of world knowledge at various times. It is given in a form which presumes a prolonged development process in concert with the environment. This can work in complex ways where the innate fragments may be creating opportunities for the necessary environmental interactions (see e.g. Sec. V-B1) or providing fragmentary representations to bootstrap the development of knowledge of objects (Sec. VI). The fact that physical knowledge is not given in a "final" form might be important to ensure that the knowledge eventually developed is linked to sensorimotor experiences of the infant, and hence more practically useful.

2) Infants Learn Slowly, but Thoroughly: Infants spend months practising individual actions in varying circumstances, and gathering good knowledge about how to apply an action, and its expected effects[17]. We see this in the way that the period dominated by affordance-based play must precede goal directed planning (Sec. II-B; see also poor planning in Sec. V-C12), and furthermore play may sometimes need to resurface when perception-action knowledge is inadequate (Sec. V-C6). Expertise and flexibility on a task come from extensive practice with the elementary actions comprising the task. During this time (in addition to the environmental circumstances varying) small variations are tried out, and the effects of those variations learned. Behaviours learnt in this slow manner are well grounded. This highlights the importance of achieving robustness and variety for controlling elementary skills, as these will come into play later when these skills may be constituents of more complex behaviour.

This slowness explains why we often see some (fortuitous) success in a particular behaviour before a fuller understanding is achieved some time later (e.g. the support, Sec. V-C3); many complex skills are learnt in a crude outline before the constituent parts are properly refined (e.g. the spoon, Sec. V-D). The early generation of experiences through this approach provides the training data which improve the behaviour.

There is a link between this general slowness and the acquisition of physical world knowledge above (Sec. VII-A1). Piaget said "to understand is to invent", and so it makes sense for the genetic "preprogramming" to only provide a fragmentary outline which guides the development of the knowledge; to achieve a thorough understanding of the physical world knowledge it is necessary for the individual to gather significant experience with the component fragments from which they can then themselves build the necessary concepts (e.g. the building of knowledge of objects, Sec. VI).When the general representations of objects and space are built in this way they are more useful because they are so closely connected to the actions that can manipulate them.

This process appears to be facilitated by a "schedule" for development which forces more time to be spent on earlier tasks; e.g. the fact that the pincer grasp arrives relatively late (Sec. V-B3) means that significant time prior to this is spent on coarser grasps, where a coarser representation of objects is adequate; see also the way perception and action may help to bootstrap each other's development (Sec. V-B5); furthermore, in language, acquisition of vocabulary proceeds very rapidly once it starts, but it does not begin until significant interaction with objects is complete.

3) Generalisation Depends on Representation: The ability to generalise and transfer to new situations is dependent on the underlying representations in use, and sometimes infants are surprisingly poor at this and seem to have knowledge that is locked in context. We have seen in Sec. VI that in some cases some of the early representations facilitate certain types of transfer (e.g. the stick), but in other cases the ability to transfer appears relatively late because it takes a long time for new appropriate representations to develop (e.g. handled objects, Sec. V-C12). The processes underlying this development are hinted at in Sec. VI, such as representational redescription (M7, see also Sec. VII-B2), but we know very little about how these processes work. They seem to be slow processes which come into play after extensive experience with more primitive context specific representations (so there is a link between this and the previous points).

Nevertheless, a lot of tool using behaviour can happen without the need for advanced representations of objects which are independent of specific tasks. Task specific learning seems to account for most observations quite well. Popular perceptions of the intellectual abilities underlying tool use sometimes overemphasise the notion of "sudden insight", and anecdotes of dramatic inventions may often turn out to have simpler explanations on closer inspection; i.e. they may be minor generalisations from very similar behaviour which was practised extensively [84, p.308],[24].

We conclude this reflection by asking what are infants good at and what are they bad at? They seem to be good at building on what they know; once they have acquired a skill, even crudely, they will try it out in varied situations, and refine it and improve it and specialise it for new situations which did not produce quite what they expected (leading to robustness and generalisation). They are good at assimilating new results and relating them to what they already know (provided there is some relation). They seem to be bad at making big leaps to new tasks that do not build on what they already know; there are tasks which are beyond them at certain ages, and it can take several months for them to acquire the necessary precursors before they can attempt them. They are however good at innovation; when presented with a task which is beyond them they will try a large range of strategies, and even if they do not succeed, they may discover something new through play.

B. Direction for Roboticists

This section offers advice for those who want to make tool-using robots which have the kind of robustness and generalisation that children have (i.e. able to cope with changes to tools and materials, and to find appropriate ways to do a job without explicit detailed instructions).

Despite our incomplete knowledge of how biological systems achieve tool use, we can outline how artificial systems might be constructed to tackle the problem in a similar way. Starting with a small set of innate sensorimotor schemas

[17]This is compatible with the principle of "developmental gradualness" [78] which describes "particular skills and abilities ... appearing initially in rudimentary forms and in highly specific contexts, and then gradually becoming more complex and wide-ranging over time".

(Sec. V-A) a bootstrapping process can be initiated by which new sensorimotor schemas develop through the interaction of innate schemas with objects in the world (Sec. V-B and V-C) by means of the six mechanisms M1–M6. In that process, the preconditions and effects of the schemas are refined and become more and more predictive. Eventually they can be utilised by a planning machinery (which is to a large degree innate) for the purpose tool use.

We believe that, when designing developing artificial cognitive systems, for some aspects it is acceptable to take artificial short-cuts, but for others one should be more careful to closely follow the biological approach. For both planning and social aspects it would seem acceptable to take advantage of the possibilities artificial systems offer; i.e. a planning system can be made available, and social demonstrations can be made directly available (through human-provided motions for example) without the need to observe or interact with a social partner. However both, the planning operators (developing from the sensorimotor schemas) and the representations of the world must develop slowly and autonomously, and *this should not be short-cut by direct coding*. To emulate this development it is valuable for roboticists to attempt to emulate the tasks which infants really can do; this avoids making robots do overly sophisticated things (which might lead the roboticist to use mechanisms that are inflexible and not generalisable). For example, if starting with behaviours achieved only at two years of age, one might need to code advanced representations, and thereby miss out on coding the processes which build those representations (missing out on one of the core mechanisms of development).

To emulate the two tracks of development of Fig. 1, the following is suggested.

1) Start with Few Schemas, to Get a Lot: We have seen that a small number of sensorimotor schemas when applied in the world, can lead naturally, by means of the mechanisms M1–M6, to a wide variety of schemas. The smallness of the initial set may be important to simplify the state-space exploration in early development, and the gradual process of additions may be important to allow them to be well grounded. By "well grounded" we mean that they must be refined through extensive practice in varied situations. For roboticists, this requires us to build systems that can generate large amounts of varying and meaningful experience and the patience to let the robot "play" for a long time.

2) Representations Must Develop Gradually: The cognitive architecture must allow representations to develop (Sec. VI), by processes such as representational redescription (M7), in order to facilitate generalisation and transfer. The system may use unsophisticated representations in the early stages of development (e.g. simple internal reproduction of perceptions with little abstraction). There must then be an ongoing process of upgrading the representations in use so as to capture more generic and abstract world knowledge. This is likely to require some scaffolding in the form of certain innate representational fragments which help the system to generate more sophisticated representations, as in the human case (Sec. VI). This must be a gradual process; if overly advanced representations are designed at an early stage, then there is a danger that they will be inflexible and non-extensible. For this reason we should not expect the early system to perform advanced tasks; it must spend a long time on simple tasks.

3) Interaction Between the Concrete and Abstract Tracks: A particular challenge is to establish mechanisms such as representational redescription (M7) which synchronise both the concrete and abstract tracks. This requires an ongoing modification and refinement of internal representations through the experience provided by the sensorimotor schemas and the adaptation of these schema to the restructured internal knowledge representation. This is a very complex task since it is very difficult to observe the change of internal representations. Here the establishing of processes in developing robot systems can actually help to understand such processes (for two examples and a more detailed discussion, see [13]).

4) Guiding Examples and Benchmarks for Development: We provided a general outline of the development of sensorimotor schemas of infants (see Fig. 2) as well as a number of concrete stages of development in solving certain tasks (see, e.g., the object-object behaviours of Sec. V-C). The general outline might serve as a guide for the overall developmental process to be realised, and the concrete examples can serve as benchmarks for truly cognitive behaviour in artificial agents.

Reflecting on the development of tool use in infants as outlined in this paper we have noted the crucial importance of developments in perception and action capabilities, and the seamless progression between this and the beginnings of tool use; this forces us to be keenly aware of the conceptual and technical hurdles still to be addressed in achieving the same in artificial systems. Nevertheless we believe that it will eventually be possible to design artificial systems that develop advanced and stable tool use capabilities by equipping them with (1) a small initial set of sensorimotor schemas, (2) a suitable architecture in which the mechanisms M1-M6 operate, and (3) large amounts of experiences generated by applying the sensorimotor schemas to objects in the world.

REFERENCES

[1] F. Guerin, "Learning like baby: A survey of ai approaches," *The Knowledge Engineering Review*, vol. 26, no. 2, pp. 209–236, 2011.

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

[3] L. B. Smith, "Dynamic systems, sensori-motor processes and the origins of stability and flexibility," in *Toward a unified theory of development: Connectionism and dynamic systems theories re-considered*, J. Spencer, M. Thomas, and J. McClelland, Eds. Ox. Uni. Press, 1997.

[4] K. W. Fischer, "A theory of cognitive development: The control and construction of hierarchies of skills," *Psychological Review*, vol. 87, no. 6, pp. 477–531, 1980.

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

[6] G. Forman, Ed., *Action and thought: From sensorimotor schemes to symbolic operation*. New York: Academic Press, 1982.

[7] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[8] M. Johnson, *The BODY in the MIND: The Bodily Basis of Meaning, Imagination, and Reason*. Chicago: University of Chicago Press, 1987.

[9] A. Stoytchev, "Some basic principles of developmental robotics," *IEEE Trans. Autonomous Mental Development*, vol. 1, no. 2, pp. 1–9, 2009.

[10] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[11] D. Mareschal, "Computational perspectives on cognitive development," *Wiley Interdisc. Reviews: Cog. Sci.*, vol. 1, no. 5, pp. 696–708, 2010.

[12] K. W. Fischer and R. W. Hencke, "Infants' construction of actions in context: Piaget's contribution to research on early development," *Psychological Science*, vol. 7, no. 4, pp. 204–210, 1996.

[13] N. Krüger, M. Popovic, L. Bodenhagen, D. Kraft, and F. Guerin, "Grasp learning by means of developing sensorimotor schemas and generic world knowledge," in *AISB Convention 2011; Computational Models of Cognitive Development*. AISB, 2011, pp. 23–31.

[14] 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.

[15] J. Bruner, "The growth and structure of skill," in *Mechanisms of motor skill development*, K. J. Connolly, Ed. New York: Academic Press, 1970, pp. 63–92.

[16] R. F. A. Cox and A. Smitsman, "The planning of tool-to-object relations in young children," *Developmental Psychobiology*, vol. 48, no. 2, pp. 178–186, 2006.

[17] B. Resende, E. Ottoni, and D. Fragaszy, "Ontogeny of manipulative behavior and nut-cracking in young capuchin monkeys (cebus apella): A perception-action perspective," *Developmental Science*, vol. 11, no. 6, pp. 828–840, 2008.

[18] E. Thelen, "Rhythmical behavior in infancy: an ethological perspective," *Developmental Psychology*, vol. 17, no. 3, pp. 237–257, 1981.

[19] P. Kellman and M. Arterberry, *The Cradle of Knowledge*. MIT-Press, 1998.

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

[21] A. Diamond and J. Gilbert, "Development as progressive inhibitory control of action: retrieval of a contiguous object," *Cognitive Development*, vol. 4, no. 3, pp. 223–249, 1989.

[22] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: a survey," *Connection Sci.*, vol. 15, no. 4, pp. 151–190, 2003.

[23] J. Bruner, "On voluntary action and its hierarchical structure," *International Journal of Psychology*, vol. 3, no. 4, pp. 239–255, 1968.

[24] P. Willatts, "Development of problem-solving strategies in infancy," in *Childrens Strategies: Contemporary Views of Cognitive Development*, D. Bjorklund, Ed. Lawrence Erlbaum, 1990.

[25] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.

[26] P. Mounoud and T. Bower, "Conservation of weight in infants," *Cognition*, vol. 3, no. 1, pp. 29–40, 1974.

[27] E. Thelen, G. Schöner, C. Scheier, and L. B. Smith, "The dynamics of embodiment: A field theory of infant perseverative reaching," *Behavioral and Brain Sciences*, vol. 24, pp. 1–86, 2001.

[28] M. Carpenter, K. Nagell, and M. Tomasello, "Social cognition, joint attention, and communicative competence from 9 to 15 months of age," *Monographs of the Society for Research in Child Development*, vol. 63, no. 4, pp. i–174, 1998.

[29] M. Tomasello, *The Cultural Origins of Human Cognition*. Harvard University Press, 1999.

[30] A. Meltzoff and K. Moore, "Persons and representation: why infant imitation is important for theories of human development," in *Imitation in infancy*, J. Nadel and G. Butterworth, Eds. Cambridge University Press, 1999, pp. 9–35.

[31] J. G. Bremner, *Infancy*. Cambridge, Mass. : Blackwell, 1994.

[32] E. Thelen and L. B. Smith, *A dynamic systems approach to the development of cognition and action*. MIT Press, 1994.

[33] T. Wiesel and D. Hubel, "Ordered arrangement of orientation columns in monkeys lacking visual experience," *J. Comp. Neurol.*, vol. 158, pp. 307–318, 1974.

[34] B. Chapman, M. Stryker, and T. Bonhoeffer, "Development of orientation preference maps in ferret primary visual cortex," *Journal of Neuroscience*, vol. 15, pp. 6443–6453, 1996.

[35] D. Kraft, R. Detry, N. Pugeault, E. Başeski, F. Guerin, J. Piater, and N. Krüger, "Development of object and grasping knowledge by robot exploration," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 368–383, 2010.

[36] R. S. Siegler and S. Ellis, "Piaget on childhood," *Psychological Science*, vol. 7, no. 4, pp. 211–215, 1996.

[37] Z. Chen, R. S. Siegler, and M. W. Daehler, "Across the great divide: Bridging the gap between understanding of toddlers' and older children's thinking," *Monographs of the Society for Research in Child Development*, vol. 65, no. 2, pp. i–105, 2000.

[38] R. S. Siegler, *Emerging Minds : The Process of Change in Children's Thinking*. Oxford University Press, 1996.

[39] O. Sporns and G. M. Edelman, "Solving bernstein's problem: A proposal for the development of coordinated movement by selection," *Child Development*, vol. 64, no. 4, pp. 960–98, Aug 1993.

[40] J. J. Gibson, *The Ecological Approach To Visual Perception*. Lawrence Erlbaum Associates, 1986.

[41] B. Hommel, "Perception in action: Multiple roles of sensory information in action control," *Cognitive Processing*, vol. 6, pp. 3–14, 2005.

[42] A. Murata, L. Fadiga, L. Fogassi, V. Gallese, V. Raos, and G. Rizzolatti, "Object representation in the ventral premotor cortex (area f5) of the monkey." *Journal of neurophys.*, vol. 78, no. 4, pp. 2226–2230, 1997.

[43] R. Held and A. Hein, "Movement-produced stimulation in the development of visually guided behavior," *Journal of Comparative and Physiological Psychology*, vol. 56, no. 5, pp. 872–876, 1963.

[44] D. Mareschal, M. Johnson, S. Sirois, M. Spratling, M. Thomas, and G. Westermann, *Neuroconstructivism, Vol. I: How the brain constructs cognition*. Oxford, UK: Oxford University Press, 2007.

[45] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[46] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, pp. 99–134, May 1998.

[47] R. S. Sutton, "Verification, the key to ai," 2006, unpublished document, available on author's webpage http://www.cs.ualberta.ca/∼sutton /IncIdeas/KeytoAI.html.

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

[49] H. H. Chaput, "The constructivist learning architecture: a model of cognitive development for robust autonomous robots," Ph.D. dissertation, AI Laboratory, The University of Texas at Austin, 2004.

[50] G. Stojanov, "Petitagé: A case study in developmental robotics," in *Proceedings of Epigenetic Robotics 1*, C. Balkenius, J. Zlatev, H. Kozima, K. Dautenhahn, and C. Breazeal, Eds., 2001.

[51] F. Perotto, J. Buisson, and L. Alvares, "Constructivist anticipatory learning mechanism (CALM): Dealing with partially deterministic and partially observable environments," in *Proc. of Seventh Int. Conf. on Epigenetic Robotics, Piscataway, NJ, USA*, 2007, pp. 117–127.

[52] E. Sahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Ücoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adaptive Behavior*, vol. 15, no. 4, pp. 447–472, 2007.

[53] J. Modayil and B. Kuipers, "Autonomous development of a grounded object ontology by a learning robot," in *Proceedings of the AAAI Spring Symposium on Control Mechanisms for Spatial Knowledge Processing in Cognitive/Intelligent Systems*. AAAI, 2007.

[54] P. Fitzpatrick and G. Metta, "Grounding Vision Through Experimental Manipulation," *Philos. Trans. of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 361, pp. 2165 – 2185, 2003.

[55] A. Stoytchev, "Robot tool behavior: A developmental approach to autonomous tool use," Ph.D. dissertation, College of Computing, Georgia Institute of Technology, 2007.

[56] S. Hart and R. Grupen, "Learning generalizable control programs," *IEEE Trans. Autonomous Mental Development (Accepted, to appear)*.

[57] N. K. 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*, accepted.

[58] I. C. Uzgiris and J. M. Hunt, *Assessment in infancy : ordinal scales of psychological development*. University of Illinois Press, 1975.

[59] C. von Hofsten, "An action perspective on motor development," *TRENDS in Cognitive Sciences*, vol. 8, no. 6, pp. 266–272, 2004.

[60] E. Thelen, "Rhythmical stereotypies in normal human infants," *Animal Behaviour*, vol. 27, no. 3, pp. 699–715, Dec 1979.

[61] C. K. Rovee and D. T. Rovee, "Conjugate reinforcement of infant exploratory behavior," *Journal of Experimental Child Psychology*, vol. 8, no. 1, pp. 33–39, 1969.

[62] P. Rochat, "Self-perception and action in infancy," *Experimental Brain Research*, vol. 123, no. 1/2, pp. 102–109, 1998.

[63] T. G. R. Bower, *Development in Infancy*. San Francisco : W.H. Freeman, 1982.

[64] A. Streri and J. Féron, "The development of haptic abilities in very young infants: From perception to cognition," *Infant Behavior and Development*, vol. 28, no. 3, pp. 290–304, Sep 2005.

[65] F. Wörgötter, A. Agostini, N. Krüger, N. Shyloa, and B. Porr, "Cognitive agents – a procedural perspective relying on the predictability of object-action-complexes (OACs)," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 420–432, 2009.

[66] C. von Hofsten, "Eye-hand coordination in the newborn," *Developmental Psychology*, vol. 18, no. 3, pp. 450–461, 1982.

[67] J. Bruner, "Organization of early skilled action," *Child Development*, vol. 44, no. 1, pp. 1–11, 1973.

[68] E. Thelen, D. Corbetta, K. Kamm, J. P. Spencer, K. Schneider, and R. F. Zernicke, "The transition to reaching: Mapping intention and intrinsic dynamics," *Child Development*, vol. 64, no. 4, pp. 1058–1098, 1993.

[69] D. C. Witherington, "The development of prospective grasping control between 5 and 7 months: A longitudinal study," *Infancy*, vol. 7, no. 2, pp. 143–161, Apr 2005.

[70] C. von Hofsten, "Developmental changes in the organization of prereaching movements," *Developmental Psychology*, vol. 20, no. 3, pp. 378–388, 1984.

[71] A. Gordon, "Development of the reach to grasp movement," in *Insights into the reach to grasp movement, Advances in Psychology, 105*, K. Bennett and U. Castiello, Eds. Elsevier, 1994.

[72] R. K. Clifton, D. W. Muir, D. H. Ashmead, and M. G. Clarkson, "Howinfants use vision for grasping objects," *Child Development*, vol. 64, no. 4, pp. 1099–1110, 1993.

[73] J. Fagard, "Linked proximal and distal changes in the reaching behavior of 5- to 12-month-old human infants grasping objects of different sizes," *Infant Behavior and Development*, vol. 23, no. 3–4, pp. 317–329, Mar 2000.

[74] C. V. Hofsten and S. Fazel-Zandy, "Development of visually guided hand orientation in reaching," *Journal Of Experimental Child Psychology*, vol. 38, no. 2, pp. 208–219, 1984.

[75] M. E. McCarty, R. K. Clifton, D. H. Ashmead, P. Lee, and N. Goubet, "Howinfants use vision for grasping objects," *Child Development*, vol. 72, no. 4, pp. 973–987, 2001.

[76] B. C. L. Touwen, "A study on the development of some motor phenomena in infancy," *Developmental Medicine and Child Neurology*, vol. 13, pp. 435–446, 1971.

[77] C. P. Johnson and P. A. Blasco, "Infant growth and development," *Pediatr. Rev.*, vol. 18, pp. 224–242, 1997.

[78] E. W. Bushnell and J. P. Boudreau, "Motor development and the mind: The potential role of motor abilities as a determinant of aspects of perceptual development," *Child Development*, vol. 64, no. 4, pp. 1005–1021, 1993.

[79] H. Ruff, "Role of manipulation in infants: responses to invariant properties of objects," *Developmental Psychology*, vol. 18, no. 5, pp. 682–691, 1982.

[80] E. W. Bushnell and J. P. Boudreau, "Exploring and exploiting objects with the hands during infancy," in *The Psychobiology of the Hand*, K. Connolly, Ed. Cambridge University Press, 1998, pp. 144–161.

[81] T. Striano and E. W. Bushnell, "Haptic perception of material properties by 3-month-old infants," *Infant Behavior and Development*, vol. 28, no. 3, pp. 266–289, Sep 2005.

[82] K. S. Bourgeois, A. W. Khawar, S. A. Neal, and J. J. Lockman, "Infant manual exploration of objects, surfaces, and their interrelations," *Infancy*, vol. 8, no. 3, pp. 233–252, 2005.

[83] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing—from humans to humanoids," vol. 26, pp. 1–20, 2010.

[84] V. Roux and B. Bril, Eds., *Stone Knapping: The necessary conditions for a uniquely hominin behaviour*. McDonald Institute Monograph Series. London: Cambridge, 2005.

[85] P. Willatts, "pulling a support to retrieve a distant object," *Developmental Psychology*, vol. 35, pp. 651–667, 1999.

[86] E. Bates, V. Carlson-Luden, and I. Bretherton, "Perceptual aspects of tool-using in infancy," *Infant Behavior and Development*, vol. 3, no. 3, pp. 181–190, 1980.

[87] B. Koslowski and J. S. Bruner, "Learning to use a lever," *Child Development*, vol. 43, no. 3, pp. 790–799, 1972.

[88] M. Kimmerle, L. A. Mick, and G. F. Michel, "Bimanual role-differentiated toy play during infancy," *Infant Behavior and Development*, vol. 18, no. 3, pp. 299–307, 1995.

[89] S. J. Lederman and R. L. Klatzky, "Hand movements: A window into haptic object recognition," *Cognitive Psychology*, vol. 19, no. 3, pp. 342–368, Sep 1987.

[90] P. R. Zelazo and R. B. Kearsley, "The emergence of functional play in infants: Evidence for a major cognitive transition," *Journal of Applied Developmental Psychology*, vol. 1, pp. 95–117, 1980.

[91] L. Fenson, J. Kagan, R. B. Kearsley, and P. R. Zelazo, "The developmental progression of manipulative play in the first two years," *Child Development*, vol. 47, no. 1, pp. 232–236, Mar 1976.

[92] A. L. Brown, "Domain-specific principles affect learning and transfer in children," *Cognitive Science*, vol. 14, no. 1, pp. 107–133, 1990.

[93] H. Örnkloo and C. von Hofsten, "Fitting objects into holes: On the development of spatial cognition skills," *Developmental Psychology*, vol. 43, no. 2, pp. 404–416, 2007.

[94] S. Street, K. James, S. Jones, and L. Smith, "Vision for action in toddlers: The posting task," *Child Development*, to appear.

[95] M. E. McCarty, R. K. Clifton, and R. R. Collard, "Problem solving in infancy: The emergence of an action plan," *Developmental Psychology*, vol. 35, no. 4, pp. 1091–1101, 1999.

[96] M. E. McCarty and R. Keen, "Facilitating problem solving performance among 9- and 12-month-old infants," *Journal of Cognition and Development*, vol. 6, no. 2, pp. 209–228, 2005.

[97] M. E. McCarty, R. K. Clifton, and R. R. Collard, "The beginnings of tool use by infants and toddlers," *Infancy*, vol. 2, no. 2, pp. 233–256, 2001.

[98] K. Connolly and J. Dalgleish, "The emergence of tool using skills in infancy," *Developmental Psychology*, vol. 25, no. 6, pp. 894–912, 1989.

[99] ——, "Individual patterns of tool use by infants," in *Motor Development in Early and Later Childhood: Longitudinal Approaches*, A. Kaverboer, B. Hopkins, and R. Geuze, Eds. European Science Foundation–Cambridge University Press, 1993, pp. 174–204.

[100] B. Achard and C. von Hofsten, "Development of the infant's ability to retrieve food through a slit," *Infant and Child Development*, vol. 11, no. 1, pp. 43–56, 2002.

[101] A. L. Brown, M. J. Kane, and C. H. Echols, "Young childrens mental models determine analogical transfer across problems with a common goal structure," *Cognitive Development*, vol. 1, pp. 103–121, 1990.

[102] K. S. Rosengren, I. T. Guti/'errez, K. N. Anderson, and S. S. Schein, "Parental reports of children's scale errors in everyday life," *Child Development*, vol. 80, no. 6, pp. 1586–1591, 2009.

[103] J. Kaufman, D. Mareschal, and M. H. Johnson, "Graspability and object processing in infants," *Infant Behavior and Development*, vol. 26, no. 4, pp. 516–528, Dec 2003.

[104] K. C. Soska and S. P. Johnson, "Development of three-dimensional object completion in infancy," *Child Development*, vol. 79, no. 5, pp. 1230–1236, Sep 2008.

[105] L. B. Smith, "From fragments to geometric shape: Changes in visual object recognition between 18- and 24- months," *Current Directions in Psychological Science*, vol. 18, no. 5, pp. 290–294, Oct 2009.

[106] M. Minsky, "Logical versus analogical or symbolic versus connectionist or neat versus scruffy," *AI Mag.*, vol. 12, pp. 34–51, April 1991.

Frank Guerin obtained his Ph.D. degree from Imperial College, London, in 2002. Since August 2003, he has been a Lecturer in Computing Science at the University of Aberdeen. He is interested in infant sensorimotor development, especially means-end behaviour and precursors to tool use. Dr. Guerin is a member of The Society for the Study of Artificial Intelligence and Simulation of Behaviour, where he has served as a committee member and co-chair of the annual convention.

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. He is leading the Cognitive Vision Lab which is focussing on computer vision and cognitive systems, in particular the learning of object representations in the context of grasping.

Dirk Kraft obtained a diploma degree in computer science from the University of Karlsruhe (TH), Germany in 2006 and a Ph.D. degree from the University of Southern Denmark in 2009. He is currently an assistant professor in the Mærsk McKinney Møller Institute, University of Southern Denmark. His research interests include cognitive systems, robotics and computer vision.

The complexity and potential of dexterous grasping examplified

Jimmy A. Jorgensen
Maersk Mc-Kinney Moller Institute
University of Southern Denmark

Dirk Kraft
Maersk Mc-Kinney Moller Institute
University of Southern Denmark

Justus Piater
Institute of Computer Science
University of Innsbruck

Henrik G. Petersen
Maersk Mc-Kinney Moller Institute
University of Southern Denmark

Norbert Krüger
Maersk Mc-Kinney Moller Institute
University of Southern Denmark

***Abstract*— We give results of exhaustive dynamic grasp simulations for 3 objects in two different scenarios – one unconstrained 'free-floating' and one constrained by a 'table' – with a standard industrial 2-finger gripper and a dexterous hand. Based on the simulation we give a quantification for the potential of grasping with this dexterous hand compared to grasping with a more simple gripper. We then investigate and quantify the transfer of successful grasps across the two kinds of scenario by means of the Matthews correlation coefficient. General consequences for dexterous grasping are drawn, in particular on the transferability and context dependency of dexterous grasping processes.**

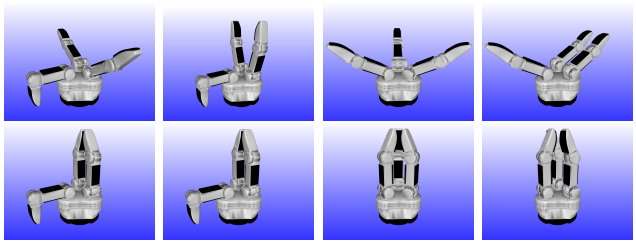


Fig. 1. Four preshapes used with the SDH-2 dexterous hand. From left to right: c_{par}, $c_{parsmall}$, c_{ball}, c_{cyl}

I. Introduction

Dexterous hands have the advantages – compared to simpler two- or three-finger hands – of being able to realize a much larger variety of grasp and manipulation options. In an industrial context this potentially allows for replacing the need to design specialized grippers for specific tasks and hence to reduce costs significantly. However, dexterous hands have not yet been used a lot in industrial applications. There are a number of reason for this such as limitedavailability of dexterous hands, the robustness of existing hardware and the cost of such devices. Fortunately in the last decade quite a number of dexterous hands have been developed and are used in labs such that the step towards industrial applications might not be so big anymore. However, another obstacle in the way of using dexterous hands is the complexity of their control and the complexity of defining suitable grasp and manipulation actions associated to specific objects.

The possible options of grasping objects with dexterous hands are virtually infinite. The success of a specific grasp applied to a specific object depends on the object shape, the grasp type, the preshape pose and the control strategy as well as the grasp context. For dexterous hands a large set of grasp types can be generated (see Figure 1) and numerous control strategies can be applied controlling forces, velocities and joint configuration. As our results indicate, even small differences in such parameters can in fact be decisive of grasp success. Moreover, there are common situations where only very specific grasp configurations lead to success. For example, when a flat object lies on a table the gripper pose needs to be controlled very carefully since a small deviation can decide between success and failure . Therefore it would be advantageous to not only code the grasps that 'mostly work' but actually the complete set of grasp affordances associated to an object to be able to choose the applicable grasps in different contexts. It is this variety of options leading to a high-dimensional search space as well as the need to choose the applicable grasps from this large variety in a context-specific manner which makes dexterous grasping a very hard task.

A naive approach would be to compute the success likelihood for each joint/force and velocity configuration at each pose associated to an object. This is infeasible due to the dimensionality of the search space. In this paper we make a compromise between coding complete grasp affordance representations and feasibility: We select a discrete set of grasp types and control strategies (see Figure 1) and compute the full set of affordances by means of simulation in a 'free floating' environment (see Figure 4 a,d,g). We then look at two different grasp contexts, namely the rather artificial free-floating context – which is often used as a context-free first test of grasps – and the application-oriented, rather common context where the object is placed on a table (see Figure 4 b,c,e,f,h,i). In this way we are able to provide insight into the parameters relevant for grasping with dexterous hands as well as the huge potential such hands provide.

We provide simulation results in terms of complete set of affordances for 3 objects (see Figure 2) using 4 different grasp types for the SDH-2 hand (see Figure 3) – one grasp type with 2 control strategies – in three different contexts ('free floating' and two 'table' environments) as shown in Figure 4. To compare with a simpler hand we also do simulations with the PG 70 two-finger hand (see Figure 3). We analyze the full set of grasp affordances for these 45 different contexts and analyze differences which provide valuable insight into grasping with dexterous hands. In particular we

R1) exemplify the superiority of dexterous hands compared to a standard 2 finger gripper (PG 70) in terms of the extend and variability of options of successful grasps to choose from. This is done by comparing grasp success distributions for these different gripper types in the different contexts.

R2) show that the likelihood of a 'random' grasp being successful is very different for different objects, grasp contexts and control strategies. We show that grasps simulated in a free-floating context are only to a small degree directly transferable to typical table picking scenarios, which motivates the need of transfer functions that are able to transfer grasp affordances from unconstrained environments (as the free floating environment) to more specific environments (such as the table environment).

R3) introduce a rather simple transfer function which allows us to transfer grasp affordances from the less constrained, free-floating environment to the table environment, and we analyze the success of this transfer which varies with the specific object and the context it is embedded in.

In this way we illustrate the potential of grasping with dexterous hands (R1),reason on the complexity of the transfer of grasping across scenarios which, as we show, depends on many factors such as the nature of the scenarios, the objects, the grippers and gripper configurations (R2), and finally present a simple method for transferring from a generic grasp representation in a 'free-floating' scenario to a specific context ('table') (R3).

In Section II we introduce the objects, grippers and preshapes used in the simulations. In Section III, we describe the simulation set-up and in Section IV the actual results which are then analyzed in Section V.

II. Selection of Objects and Grippers

Three objects were selected from the KIT Object-Models Web Database[1], see Figure 2. The objects are common household objects which are sufficiently different to provide interesting comparisons.

The grippers used to grasp these objects are the Schunk parallel gripper (PG 70) and the Schunk Dexterous Hand (SDH-2), see Figure 3. The PG 70 gripper has two fingers coupled into one Degree Of Freedom (DOF), that is, 1 DOF moves both fingers. The fingers can move up to 7 cm apart and the contact surface is approximately 2x3cm and covered by rubber.

[1]http://wwwiaim.ira.uka.de/ObjectModels

Fig. 2. From left to right: corny object, cup-object and tomato object from the KIT object database

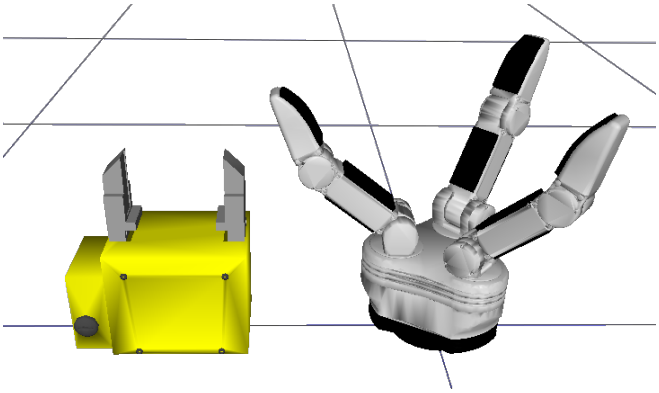


Fig. 3. PG 70 gripper (left) and the SDH-2 (right).

The SDH-2 is a 3-fingered dexterous hand with 2 DOF per finger and one coupled DOF to control the base rotation of two of the three fingers. The SDH-2 has 6 contacting surfaces covered with rubber, each of them measuring approximately 2×3 cm. However, for precision grasps only the 3 contact surfaces on the distal joints are normally used.

A number of preshapes can be associated to each gripper. Being a simple parallel gripper the PG 70 only has one preshape $q = 3.4\,\mathrm{cm}$, giving it an maximum distance of 6.8cm between its jaws. Four preshapes were chosen for the SDH-2 which are shown in Figure 1. These different preshapes enable different grasping options, and as such are important when characterizing the grasp affordances of the gripper.

We need to use preshapes because of multiple properties:

- Using preshapes is a simple and direct way of providing multiple ways of grasping an object. The process reduces to the sequence: *(open hand, move toward object, close hand)*, which is already supported by the software of the SDH-2. Alternatively, complex grasp planners that rely on sensor data are typically needed.
- Preshapes do not require additional parametrization. This property is important when sampling random grasps since the dimensionality of the search space is not extended by any gripper parameters such as the individual joint configurations.
- Using preshapes enables an easy comparison between the grippers since the dexterous gripper reduces to a

simple open/close gripper for each preshape. Hence, the same analysis can be applied.

We perform exhaustive grasp simulations for the three objects and all pre-shapes associated to the SDH-2 hand and the PG70 in two kinds of scenarios: and

- *free-float* – In the free-floating scenario the gravity is set to zero, and gripper and object are the only geometries in the simulation.
- *on table* – In the table scenario the gravity is set to $9.8\,\mathrm{m/s^2}$ and a table (plane) is added to the environment. The object may rest on this table in several different poses depending on object shape, so several table scenarios per object are used, see Figure 4.

III. Computation of Grasp Affordances

In this paper, we compute complete grasp affordances by evaluating randomly sampled gripper poses in the neighbourhood of an object. Each of these sampled poses, combined with a preshape of the gripper, represents a grasp hypothesis which can be evaluated in simulation. The outcome of the simulation may be one of *(success, failure or collision)*, where *success* represents a successful grasp of the object (indicated by the fact that the fingers are still in contact with object after grasping it) and *failure* represents a grasp where the gripper has no contact with the object after trying to grasp it; *collision* represents a grasp which is initially in collision with the object or the environment.

The evaluation of a grasp hypothesis is performed using a dynamics grasp simulator from RobWork [2]. The main simulation process of a single grasp is:

1) Set the initial scene configuration, eg. gravity, friction, pose of obstacles and objects.
2) Place gripper in a "random" pose (relative to the object) and set the gripper configuration to one of the preshapes.
3) Test if the gripper collides with object or environment.
4) Start the simulation and set the target gripper configuration (preshape dependent) of the gripper controller.
5) If a grasp is obtained, lift the object and compute the success criteria.

The sampling of the gripper and the choice of sampling strategy necessarily influence the resulting set of grasp affordances and the overall success probability. Typically grasp planning focuses on gaining a high overall success probability by exploiting knowledge of gripper, object and environment. However, in this work we focus on the analysis and comparison of comprehensive grasp affordances; thus, an unbiased sampling strategy is preferred. Uniformly random sampling in $SE(3)$ would therefore be ideal but is impractical because of the large number of simulations necessary to cover $SE(3)$. Instead a sampling strategy that is biased toward the object geometry is used. This effectively reduces the number of required simulations without biasing the success probability toward a specific gripper. The overall success probabilities can then be used as a measure of the volume of the subspace of grasp successes in SE(3) which can be used as a quantification in the context of R1.

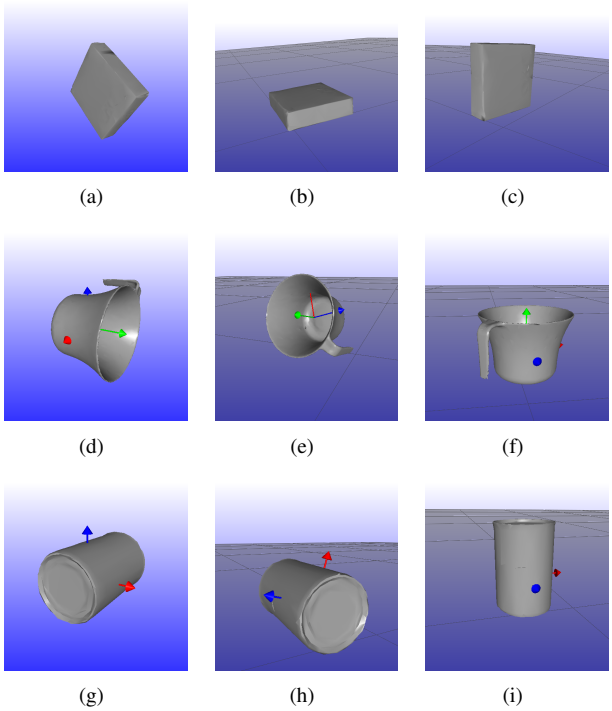


Fig. 4. The floating pose is illustrated on the left, followed by two canonical poses on the table (side and upright). Top row: Corny object. Middle row: Cup object. Bottom row: Tomato object.

A set of grasp affordances are defined for a specific object, with a specific gripper and preshape, in one of the two specific contexts, free floating or placed on a table.

Two canonical poses of each object in the table environment are used to provide different grasp contexts. These are illustrated in Figure 4 together with the free-floating context.

A. Object specific sampling strategy

The sampling strategy used for sampling the pose of the gripper relative to the object primarily follows the object geometry. However, it also assumes that the approach vector of the gripper is placed in the same direction as the positive z axis of the gripper Tool Center Point (TCP) frame. It effectively encapsulates the idea that the gripper needs to point toward some part of the object geometry before it is able to successfully grasp it.

First a random point $\mathbf{p}$ on the surface of the object is selected. Then a uniformly distributed orientation $\mathbf{R}$ in $SO(3)$ is calculated and used to define the temporary target pose $\mathbf{T_{tmp}} = (\mathbf{p}, \mathbf{R})$. The pose is then translated along the z axis by a randomly generated value d in the interval $[-0.04\,\mathrm{m}; 0.04\,\mathrm{m}]$. The final pose is therefore:

$$\mathbf{T_{pose}} = (\mathbf{p} - (\mathbf{R} \cdot [0,0,1]^T) \cdot d, \mathbf{R}) \quad (1)$$

As can be seen the strategy does not directly take into account properties of the different grippers and as such does not explicitly bias the sampling toward a specific gripper.

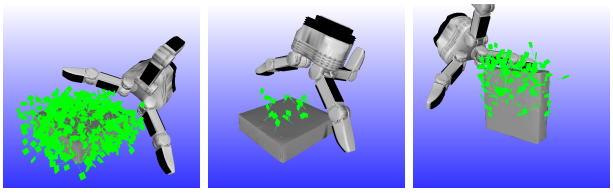


Fig. 5. Each show the successes of grasping the Corny object in the same 5000 grasp simulations but performed in different contexts. From left to right: free-float, on table (side pose) and on table (upright pose).

B. Transfer function

The sampling strategy is used once per object to generate a grasp hypothesis for the floating environment. The same grasp hypotheses are then also used for the *on table* environments. This enables the comparison between the outcome of grasp hypotheses in the table environments against the floating environment, which is necessary to answer how well the outcome of the floating environment predicts the outcome in the table environments.

A large proportion of successful grasps in the free-floating environment will be in collision in the table environment because of the added table geometries. Naively predicting success in the table environment from success in the floating environment will therefore not suffice. We therefore use a simple transfer function to remove any grasps where the gripper is not above the table. In practice this is simply calculated by checking if the gripper collides with the table plane.

IV. Simulation Results

Figure 5 illustrates the successful outcomes of the same 5000 grasp hypotheses in the three different contexts of the corny object grasped with the SDH-2 with the preshape c_{par}. It is clear that the added table constraint significantly reduces the number of successful grasps. However, it is not clear if successes from the constrained environments (center and right image) will also be successes in the floating environment. In the following we use confusion matrices to evaluate how well successes and failures in the floating simulations predict successes and failures in the table environments, and vice versa, as illustrated in Table I.

Multiple datasets were generated for each gripper. For the SDH-2 the datasets are characterized by a triple (o_i, s_j, c_k), where o_j is the object, s_j is the specific scene (free floating or on table, (with different poses), and c_k is the grasp strategy which includes the number of fingers and the preshape used. The parallel gripper is simpler, and only one grasp strategy is used. Hence we describe datasets generated with the PG 70 by a pair (o_i, s_j).

For each floating environment (o_i, s_{float}, c_k), 100.000[2] grasp simulations were generated using the sampling approach presented in Section III-A. The overall success probabilities of the simulations are available in Table II. Two success probabilities are given. The first shows the percentage of grasp successes from all grasps that were not initially in collision, the second shows the percentage of grasp successes from all grasps including the colliding ones.

In the same table the success probabilities of the simulations of the table environments (s_{side}, s_{uprigt}) are also shown. These simulations use the same grasp hypotheses as in the floating environment but performed in the specific contexts (e.g. the object was placed upright on the table). This makes it possible to compare the predictability of the floating environment simulations, that is, whether a grasp successful in the floating environment is also successful in the specific table context, and vice versa. This predictability is quantified by the confusion matrices in Table III, as illustrated in Table I.

SDH-2 o_{corny}, c_{ball}		Results in s_{float}	
		success	failure
Results for s_{side}	success	TP=74 (true positives), the number of grasp simulations that where successfull in both scenarios	FN=113 (false negatives), the number of grasp simulations that failed in s_{float} but succeeded in s_1
	failure	FP=539 (false positives), the number of grasp simulations that succeeded in s_{float} but failed in s_1	TN=2507 (true negatives), the number of grasp simulations that failed in both s_{float} and s_1

TABLE I

EXAMPLE OF A CONFUSION MATRIX ELEMENT FROM THE SUCCESS COMPARISON TABLES. HIGH VALUES IN TP AND TN INDICATE HIGH TRANSFERABILITY OF GRASP RESULTS FROM ONE SCENARIO TO THE OTHER.

V. Analysis of Results

We give measures supporting the questions R1, R2 and R3 posed in the introduction.

A. Superiority of dexterous grasping (R1)

The success probabilities visualized in the previous section show a clear picture of the superiority of the dexterous gripper compared to the PG 70 in our setting. We express the difference between the potential of the two-finger gripper PG 70 and the SDH-2 for an object o and a context c by the measure

$$d^{(o,c)}(PG\ 70, SDH) = \frac{\max(c_{par}, c_{parsmall}, c_{ball}, c_{cyl})}{c_{0,PG\ 70}} \tag{2}$$

The values of $d^{(o,c)}(PG\ 70, SDH)$ are given on the right in Table II.

It can be concluded that the grasp success probability for the SDH-2 (under the condition that a suitable preshape is selected) exceeds the success probability by a factor of 1.4 to 380. On average (over all scenarios and objects) this factor is 88 in our experiments. Even with a rather small set of preshapes tested and assuming an ideal choice of preshape, grasping with the dexterous hand provides a significantly higher chance of grasp success compared to

[2]For the tomato object the actual number of samples is slightly lower.

		SDH-2								PG 70		$d^{(o,c)}$
		c_{par}		$c_{parsmall}$		c_{ball}		c_{cyl}		c_0		
o_{corny}	s_{float}	43.1%	25.0%	45.5%	5.2%	70.3%	38.0%	52.9%	28.0%	12.3%	0.7%	54.2
	s_{side}	5.8%	0.4%	0.0%	0.0%	5.8%	0.2%	5.3%	0.3%	0.1%	0.0%	380.0
	$s_{upright}$	42.1%	7.2%	41.5%	1.9%	70.9%	9.3%	51.9%	8.9%	14.4%	0.4%	25.1
o_{cup}	s_{float}	54.5%	42.6%	47.7%	6.0%	79.8%	61.1%	61.7%	45.7%	41.5%	3.7%	16.5
	s_{side}	47.0%	5.8%	48.4%	1.7%	72.5%	5.4%	47.3%	6.0%	44.0%	1.6%	3.6
	$s_{upright}$	45.0%	4.7%	65.4%	2.3%	81.9%	4.9%	47.7%	4.9%	70.4%	3.5%	1.4
o_{tomato}	s_{float}	48.5%	24.9%	30.4%	5.5%	84.9%	72.0%	72.3%	59.9%	3.8%	0.3%	240.0
	s_{side}	22.7%	2.3%	10.9%	0.3%	34.4%	1.9%	32.8%	3.4%	2.4%	0.1%	52.3
	$s_{upright}$	43.4%	6.6%	23.1%	1.4%	75.3%	7.5%	63.1%	10.4%	17.2%	0.6%	18.9

TABLE II

SUCCESS PERCENTAGES OF THE SIMULATED OUTCOMES. IN EACH MAJOR COLUMN, THE LEFT SUBCOLUMN SHOWS THE PERCENTAGES OF SUCCESS IF COLLISIONS ARE NOT INCLUDED, AND THE RIGHT COLUMN SHOWS THE SUCCESS PERCENTAGES IF COLLISIONS ARE INCLUDED. THE RIGHTMOST COLUMN SHOWS THE VALUES OF THE R1 METRIC (2) INTRODUCED IN SECTION V-A.

		SDH-2								PG 70	
		c_{par}		$c_{parsmall}$		c_{ball}		c_{cyl}		c_0	
o_{corny}	s_{side}	159	221	0	0	74	113	97	222	0	1
		933	5298	1	1495	539	2507	721	4991	0	1499
	$s_{upright}$	6271	861	1777	100	8586	761	6296	830	327	46
		1660	8183	284	2379	1127	2712	1587	5017	34	2182
o_{cup}	s_{side}	4876	916	1548	126	1459	407	893	314	1568	81
		1903	4621	241	1546	303	406	474	871	175	1924
	$s_{upright}$	3483	1219	2093	207	1574	657	917	540	3119	396
		1683	4059	155	1061	195	297	449	1146	78	1399
o_{tomato}	s_{side}	1019	228	246	31	445	112	1063	278	30	35
		1310	2923	214	2045	570	494	1212	1531	43	2568
	$s_{upright}$	1518	119	547	82	2167	91	2019	112	113	434
		538	1591	219	1874	261	477	607	602	53	2584

TABLE III

CONFUSION MATRICES OF SUCCESSES AND FAILURES FROM FLOATING ENVIRONMENT AND THE SPECIFIC TABLE ENVIRONMENT ($s_{side}, s_{upright}$). SEE TABLE I FOR A DETAILED EXPLANATION OF A SINGLE CELL.

		SDH-2				PG 70
		c_{par}	$c_{parsmall}$	c_{ball}	c_{cyl}	c_0
o_{corny}	$s_{side} MCCR2$	-0.0021	NAN	-0.040	-0.035	-0.0051
	$s_{side} MCCR3$	0.17	NAN	0.13	0.12	NAN
	$s_{upright} MCCR2$	0.34	0.44	0.22	0.31	0.61
	$s_{upright} MCCR3$	0.70	0.83	0.64	0.65	0.87
o_{cup}	$s_{side}, MCCR2$	0.17	0.35	-0.012	0.077	0.51
	$s_{side} MCCR3$	0.55	0.79	0.34	0.39	0.86
	$s_{upright} MCCR2$	0.10	0.41	-0.063	0.0090	0.78
	$s_{upright} MCCR3$	0.45	0.78	0.25	0.35	0.79
o_{tomato}	$s_{side} MCCR2$	0.11	0.16	-0.021	0.032	0.20
	$s_{side} MCCR3$	0.43	0.64	0.26	0.33	0.42
	$s_{upright} MCCR2$	0.25	0.35	0.096	0.19	0.26
	$s_{upright} MCCR3$	0.67	0.72	0.67	0.52	0.32

TABLE IV

QUALITY ESTIMATES OF THE CONFUSION MATRICES IN TABLE III.

the PG 70 in 43 of 45 contexts by as much as an order of magnitude. *This indicates the availability of a significantly larger volume of successful grasping option of dexterous hands compared to standard industrial grippers. This is important when grasping is done in constrained scenarios in which only few grasp options might be executable due to limitation of workspace of the robot as well as complex contexts (e.g., when the object is situated in a pile of objects, as e.g. in a bin–picking context). However, it comes with the cost of needing to select an appropriate control strategy.*

B. Direct transfer between free-floating scenario and table scenario (R2)

We express the success of a direct transfer as well as the transfer function introduced in section III-B for both PG 70 and SDH-2 by the Matthews correlation coefficient (MCC) [1]. The MCC produces a value between -1 and 1, where 1 indicates perfect prediction, 0 indicates random prediction

and -1 indicates inverse prediction.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (3)$$

In the direct-transfer scenario, we naively apply a successful grasp from the free floating scenario to the more constrained table scenarios.

The MCC values for the direct-transfer scenario are given as every other row (labeled $MCCR2$) in Table IV. Some of these values are very small, notably in those cases where the object lies flat on the table (corresponding to the rows labeled $s_{side}MCCR2$) and the table surface impedes most grasping approaches. Nevertheless, most of the other cases are better behaved, and the MCC values indicate that *transfer of grasp parameters from free-floating to more constrained scenarios is possible to some degree*.

C. Simple transfer function between free-floating scenario and table scenario ($R3$)

We measure the success of the simple transfer function introduced in section III-B for both the PG 70 and the SDH-2 using the same MCC measure (3) above. The metric is used directly on the confusion matrices of Table III (see table I for explanation). Note that in table III collisions to the table are filtered away (i.e., the transfer function introduced in section III-B is applied. The results are shown as every other row (labeled $MCCR3$) in Table IV. The results show that the quality of the prediction increases substantially compared to the direct transfer (shown in the respective preceding rows). In most contexts the MCC is much increased with values as high as 0.80, indicating a very good transfer even by our rather simple transfer function. Even for the difficult cases of objects lying sideways on the table, most MCC values increase, even though some remain very low nevertheless.

It is important to mention that in certain contexts quite a number of grasps that are not successful in the free-floating environment are succesfull in the constrained environment. This implies that in certain scenarios context-specific control strategies need to be taken into account. For example, the movement of a flat object on a table when touched by one of the fingers might be utilized in the grasping process. These context-specific constraints cannot be accounted for in a free-floating scenario in the same way, and need to be learned in the specific context. *In summary, the results show that grasp simulation in free-floating scenarios – using a straightforward transfer function – give in general quite reliable indications of grasp success in more constrained scenarios. However, in certain contexts better transfer functions are required*, which should ideally be learned in a context-specific fashion.

VI. Conclusion

In large-scale simulations for two grippers and three objects in different contexts, we investigated and quantified the potential of dexterous grasping compared to grasping with a two finger gripper, indicating a large potential of dexterous grippers (R1). We investigated the transfer between an unconstrained, free-floating environment and more constrained environments, which is important for applying learned grasp knowledge in novel contexts. We showed that grasp success likelihoods depend heavily on the context the object is embedded in, and that a naive transfer from a free-floating scenario only makes sense in some contexts (R2). We then defined a rather simple transfer function – which basically takes collision constraints into account – and showed that such a transfer function can give rather good results in constrained contexts (R3). The results also suggest that in some cases context-specific learning of control strategies is desirable.

References

[1] B.W. and Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451, 1975.

[2] J. Jorgensen, L. Ellekilde, and H. Petersen. Robworksim - an open simulator for sensor based grasping. In *Proceedings of Joint 41st International Symposium on Robotics (ISR 2010) and the 6th German Conference on Robotics*, Munich, 2010.

Object-Action Complexes: Grounded Abstractions of Sensorimotor Processes

Norbert Krüger[a], Christopher Geib[b], Justus Piater[c], Ronald Petrick[b], Mark Steedman[b], Florentin Wörgötter[d], Aleš Ude[e], Tamim Asfour[f], Dirk Kraft[a], Damir Omrčen[e], Alejandro Agostini[g], Rüdiger Dillmann[f]

[a]*Mærsk McKinney Møller Institute, University of Southern Denmark, Odense, Denmark*
[b]*School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK*
[c]*Institute of Computer Science, University of Innsbruck, Austria*
[d]*Bernstein Center for Computational Neuroscience (BCCN), Göttingen, Germany*
[e]*Jožef Stefan Institute, Department of Automatics, Biocybernetics, and Robotics, Ljubljana, Slovenia*
[f]*Institute for Anthropomatics (IFA), Humanoids and Intelligence Systems Laboratories (HIS), Karlsruhe Institute of Technology, Karlsruhe, Germany*
[g]*Institut de Robotica i Informatica Industrial (CSIC-UPC), Barcelona, Spain*

Abstract

This paper formalises Object-Action Complexes (OACs) as a basis for symbolic representations of sensorimotor experience and behaviours. OACs are designed to capture the interaction between objects and associated actions in artificial cognitive systems. This paper gives a formal definition of OACs, provides examples of their use for autonomous cognitive robots, and enumerates a number of critical learning problems in terms of OACs.

Keywords: Robotics, grounding, reasoning about action and change, execution monitoring, machine learning

1. Introduction

Autonomous cognitive robots must be able to interact with the world and reason about the results of those interactions, a problem that presents a number of representational challenges. On the one hand, physical interactions are inherently continuous, noisy, and require feedback (e.g., consider the problem of moving forward by 42.8 centimetres or until a sensor indicates an obstacle). On the other hand, the knowledge needed for reasoning about high-level objectives and plans is more conveniently expressed in a symbolic

form, as predictions about discrete state changes (e.g., going into the kitchen enables retrieving the coffee pot). Bridging the gap between low-level control knowledge and high-level abstract reasoning has been a fundamental concern of autonomous robotics [1, 2, 3, 4]. However, the task of providing autonomous robots with the ability to build symbolic representations of continuous sensorimotor experience *de novo* has received much less attention, even though this capability is crucial if robots are ever to perform at levels comparable to humans.

To address this need, this paper proposes a formal entity called an *Object-Action Complex* (OAC, pronounced "oak") as the basis for symbolic representations of sensorimotor experience. The OAC formalism is designed to achieve two ends. First, OACs provide a computational account that brings together several existing concepts from developmental psychology, behavioural and cognitive robotics, and artificial intelligence. Second, by formalising these ideas together in a shared computational model, OACs allow us to enumerate and clarify a number of learning problems faced by embodied agents. Some of these learning problems are known and have been well studied in the literature, while others have received little or no attention.

OACs are designed to formalise adaptive and predictive behaviours at all levels of a cognitive processing hierarchy. In particular, the formalism ensures that OACs are grounded in real-world experiences: all learning and refinement of OACs will be based on statistics gained from an agent's ongoing interaction with the world. To relate OACs operating at different processing levels, we will also allow OACs to be defined as combinations of other OACs in a hierarchy, in order to produce more complex behaviours. As a result, this formalism enables consistent, repeatable hierarchies of behaviour to be learnt, based on statistics gained during real-world interaction, that can also be used for probabilistic reasoning and planning. It also provides a framework that allows the OAC designer to focus on those design ideas that are essential for developing cognitive agents.

The goal of the OAC formalism is to provide a unifying framework for representing diverse interactions, from low-level reactive behaviour to high-level deliberative planning. To this end, we will build our computational models on existing assumptions and ideas that have been shown to be productive in previous research, in a number of different fields. In particular, we note six design ideas (DI) that have helped motivate our formalism:

DI-1 **Attributes:** Actions, objects, and interactions must be formalised over

an appropriate attribute space, defined as a collection of properties with sets of associated values. An agent's expectations and predictions (see [DI-2]) as to how the world will change if an action is performed must also be defined over such an attribute space. Different representations may require different attribute spaces, plus a method of mapping between them if they are to be used together.

DI-2 **Prediction:** A cognitive agent performing an action to achieve some effect must be able to predict how the world will change as a result of this action. That is, it must know which attributes of the world must hold for an action to be possible (which will typically involve reasoning about the presence of objects), which attributes will change when the action is performed, and how those attributes will change.

DI-3 **Execution:** Many previous efforts to produce fully autonomous robotic agents have been limited by simplifying assumptions about sensor, action, and effector models. We instead take the approach that complete robotic systems must be built with the ability to actually execute actions in the world and evaluate their success. This requires agents to be embodied within physical systems that can interact with the physical world.

DI-4 **Verification:** In order to improve its performance in a nondeterministic physical world, an agent must be able to evaluate the effectiveness of its actions, by recognising the difference between the states it predicted would arise from its actions, and those states that actually resulted from action execution.

DI-5 **Learning:** State and action representations are dynamic entities that can be extended by learning in a number of ways: continuous parameters can be optimised, attribute spaces can be refined or extended, new control programs can be added, and prediction functions can be improved. Embodied physical experiences characterised in terms of actions, predictions, and outcomes provide data for learning at all levels of a system.

DI-6 **Reliability:** It is not sufficient for an agent to merely have a model of the changing world. It must also learn the reliability of this model. Thus, our representations must measure and track the accuracy of their predictions over past executions.

These design ideas are widely accepted in the literature, where they have been discussed by various authors (see, e.g., [5, 6]). For example, a STRIPS-style planning operator [7] can be seen as a prediction function [DI-2] built from action preconditions and effects defined over an attribute space [DI-1]. Significant work has also been done on learning such prediction functions given an appropriate attribute space [8, 9]. The importance of embodiment [DI-3] in real-world cognitive systems has been pointed out by Brooks [1, 2]. Richard Sutton [10] has discussed the necessity of verifying the expected effects of actions [DI-4] to arrive at meaningful knowledge in AI systems. The interplay between execution [DI-3] and verification [DI-5] is associated with the *grounding problem* [11]. For example, Stoytchev [5] defines grounding as "successful verification", and discusses the importance of evaluating the success of actions [DI-6] and maintaining "probabilistic estimates of repeatability". We will discuss the relation of our work to prior work further in Section 3.

1.1. Paper Structure

In the remainder of the paper we will develop the OAC concept using the above design ideas. In particular, this paper will:

- formally define OACs for use by autonomous cognitive agents,
- identify problems associated with learning OACs, and
- provide examples of OACs and their interaction within embodied systems.

The rest of the paper is organised as follows. Section 2 further motivates this work and provides some basic terminology. Section 3 discusses the relation to prior research. Section 4 provides a formal definition of OACs, based on the above design ideas. Section 5 characterises a number of learning problems in terms of OACs. Section 6 describes how OACs are executed within a physical robot system. Section 7 provides detailed examples of OACs. Finally, Section 8 demonstrates a set of OACs interacting with each other to realise cognitive behaviour, including object grounding and associated grasping affordances, as well as planning with partly grounded entities.

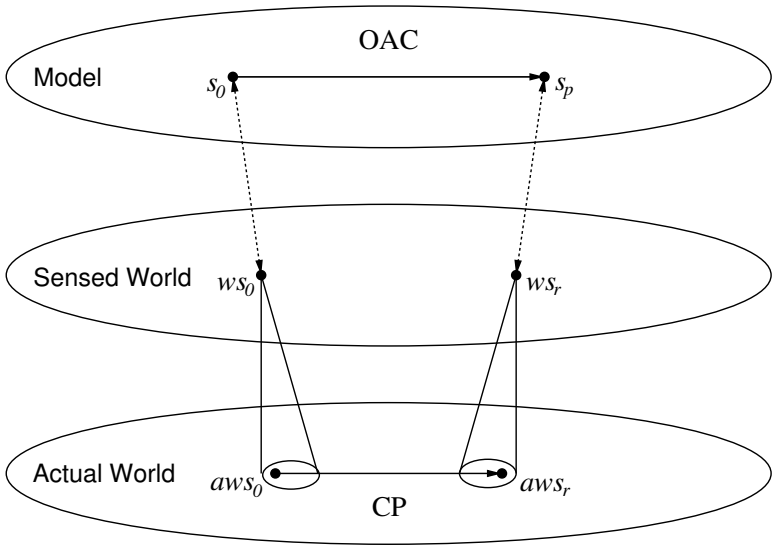


Figure 1: Graphical representation of an OAC and its relationship to a control program.

2. Prerequisites for Modelling OACs

To achieve its goals in the real world, an embodied agent must develop predictive models that capture the dynamics of the world and describe how its actions affect the world. Building such models, by interacting with the world, requires facing a number of representational challenges resulting from

- the continuous nature of the world,
- the limitations of the agent's sensors, and
- the stochasticity of real-world environments.

These problems make the task of efficiently predicting the results of real-world interactions challenging, and often require highly-specialised models of the interaction. As a result, any framework for representing such interactions must be able to support multiple models of the world, based on different attribute spaces. For example, differential equations can be straightforwardly used to predict the trajectory of straight line motions. However, this kind of representation will not be effective for symbolic planning. We will call each model of an interaction with the world an OAC, and stipulate that each OAC be defined over an attribute space.

Given the continuous nature of the world, all of an agent's interactions with the external world must be mediated by low-level continuous control routines. Such routines are necessary for the agent to sense and to act in a noisy, continuous, and uncertain real world. For this exposition, we will assume that the agent has a low-level control program (CP) that it uses to interact with the world.[1] Our objective then is for OACs to capture the interactions with the world meditated by the CPs. In order words, we will describe an OAC as *modelling* a CP.

Our first three design ideas suggest that an OAC must contain a prediction function defined on an attribute space that captures the regularities and results of its specific CP. Figure 1 illustrates this idea graphically with an OAC that predicts the behaviour of a specific CP functioning in the real world to move an agent's end effector. Here, the control program causes changes in the actual world that transform the actual initial state of the world, denoted by aws_0 (and sensed by the agent as ws_0), to the resulting actual world state, aws_r (sensed by the agent as ws_r).

An OAC that models this CP must also be able to map states of the sensed world to states represented in terms of its own attribute space, and to make predictions about the transitions that are caused by the CP. In Figure 1 this is captured by a correspondence between ws_0 in the sensed world and the initial state s_0 in the OAC's attribute space, and the OAC's predicted state s_p and the resulting sensed state ws_r.

In practice, we can simplify this diagram slightly. Since the agent's perception of the world is completely mediated by its sensors and effectors, any change in the world can only be observed by the agent through its (possibly faulty) sensors. Further, because the available sensor set of a given agent is fixed, we can treat the actual world and the sensed world as a single level, as shown in Figure 2. While we recognise the presence of errors in the sensors and the inherently un-sensed variation of the world, since we are not modelling learning over evolutionary time scales, we also assume that all embodied agents must learn based on the noisy and incomplete sensors provided to them. We will make this assumption for the remainder of the paper.

[1] We will discuss how new CPs can be learnt later in this document, but for the purpose of introducing this idea we will simply assume a CP is given.

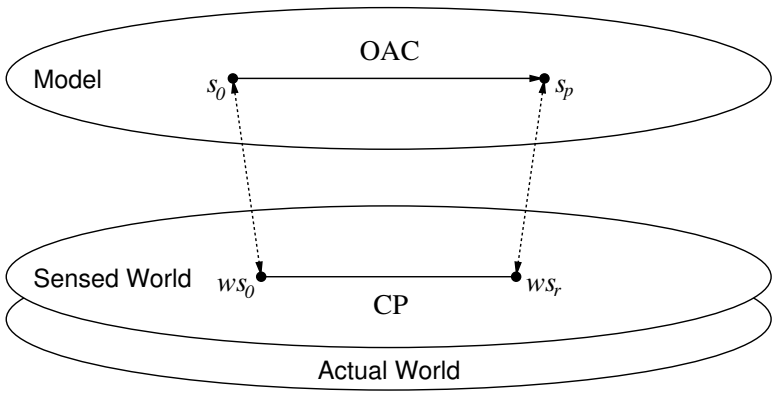


Figure 2: Graphical representation of an OAC and its relationship to the sensed world and a control program.

2.1. Representational Congruency and Grounding

For an OAC to model a CP and be effective for high-level reasoning tasks, it must consistently capture the underlying regularities present in the execution of the CP. One way to do this is to ensure that the states modelled by an OAC are inferable from sensed features of the world, and that relevant changes in the sensed world resulting from the execution of the CP are reflected in the states predicted by the OAC. We will call this property *representational congruency*, and will refer to OACs with this property as *congruent models* or *congruent representations* of the CP.

Representational congruency imposes strong conditions on an OAC's prediction function and attribute space, with respect to the CP it models, as illustrated in Figure 2. In particular, if s_0 is an initial state in the OAC's attribute space, corresponding to the sensed state ws_0, and the sensed state ws_r results from the execution of a CP in ws_0, then the state s_p predicted by the OAC (and represented in the OAC's attribute space) must map to the sensed state ws_r. In practical terms, such guarantees are necessary for ensuring the correctness of high-level reasoning tasks that have consequences at the sensed world level. (For instance, building high-level plans that are interpreted in terms of low-level effector commands.) We will provide a formal definition of representational congruency, and a further discussion, in Section 6.

We note that nothing about representational congruency requires OACs to share attribute spaces with the sensed world. For instance, a single state in the OAC's attribute space might denote a set of states at the sensed world

level. In general, a representationally congruent OAC is free to abstract its attribute space from the sensed world in any way that is effective for its reasoning task. This allows each OAC to develop and work with representations that are specific to their own reasoning tasks.

We also note that learning such congruent models requires a mapping from the sensed world state to the OAC's attribute state that is *consistent*. In other words, a given sensed state of the world must always map to the same state in the OAC's attribute space. Without such consistencies, regularities in the execution of the CP cannot be recognised, let alone learnt and modelled by an OAC. Given consistent mappings, we envision the congruency of an OAC increasing as experience extends the OAC's attribute space.

Following DI-3, DI-4, and DI-5, we will also require all OAC learning and refinement to be based on statistics gained through an agent's interaction with the world, in order to ensure that the resulting OACs are grounded in real-world execution and sensor feedback. Thus, we envision cognitive systems using OACs to solve a problem at a high level of abstraction while grounding their real-world interactions with low-level control programs and sensed world states. Further, while this section has discussed OACs as being grounded by executing a single control program, in Section 6 we will discuss how OACs can be defined as a combination of lower-level OACs. This will enable consistent, probabilistic reasoning and planning based on statistics gained during the execution of OACs in the world.

With these intuitions in hand, we will now discuss the relationship of OACs to prior work.

3. Relation to other Approaches

The OAC concept provides a *framework* for formalising actions and their effects in artificial cognitive systems, while ensuring that relevant components and prerequisites of action acquisition, refinement, chaining and execution are defined (e.g., the attribute space, a prediction of the change of the attribute space associated with an action together with an estimate of the reliability of this prediction, an execution function, and a means of verifying the outcome of an action). In particular, this framework ensures the grounding of an OAC in sensory experience by means of incremental verification and refinement ("ongoing learning"). It also specifies which components of an OAC are subject to learning as outlined in Section 5. Our OAC definition, however, does not specify the actual learning algorithms (e.g., whether this

learning takes place in a neural network, by means of reinforcement learning, or based on Bayesian statistics); this is up to the designer of the concrete OAC. As such, OACs ensure certain properties of action representation are fulfilled, leaving the designer free to specify the remaining content. The OAC framework thus provides a basis for the design of elementary cognitive units and their interaction. Naturally, it is based on a significant amount of prior work on action representations, as we will outline below.

A closely related concept from psychology is the (sensorimotor) schema as defined by Piaget and others [12, 13]. A sensorimotor schema is a dynamic entity that gathers together the perceptions and associated actions involved in the performance of behaviours. The schema represents knowledge generalised from all the experiences which have been involved in the executions of that behaviour. It also includes knowledge about the context in which the behaviour was performed as well as the agent's expectations about the action effects. Cognitive development takes place by refining and combining these schemas. OACs can be seen as a formalisation of such schemas to be used in artificial cognitive systems.

Together, the different components of OACs formalise concepts which have been derived over the last decades in cognitive science, artificial intelligence and robotics. We discuss related work in terms of four subjects that are addressed by the OAC concept: (1) the definition and learning of suitable attribute spaces and the predictions taking place in these spaces, (2) the concept of affordances, (3) the grounding of symbolic entities by the agent's interaction with the world, (4) the modularisation of actions allowing for their flexible combination, (5) hybrid control schemes, and (6) learning and memorisation.

Attributes and the prediction of expected change: The representation of world states in terms of discrete attribute spaces, and the representation of actions as expected changes to the values of these attributes, can be directly linked to STRIPS [7] and other classical formalisms [14, 15, 16]. Predictability of cause and effect (or the lack of it) is important for cognitive agents and has been treated in a large body of work [17, 18, 19, 20, 21, 22]. However, unlike classical formalisms, the prediction function associated with an OAC constitutes a dynamic and grounded entity, changing under the influence of ongoing learning processes in the cognitive system.

More specifically, OACs go beyond such classical representations by permitting both continuous and discrete attribute spaces, making it possible to

use OACs at different levels of a processing hierarchy, from low-level sensorimotor processes for robot perception and control, to high-level symbolic units for planning and language. As a consequence, OACs can be viewed as containers enabling subsymbolic as well as symbolic representations, and models of both symbolic and subsymbolic cognition can be formalised using OACs (see [23]).

Structures like POMDPs (see, e.g., [24]) are related to OACs in that they are also defined in terms of states, actions, and state transitions. However, unlike OACs, they employ specific probabilistic representations tailored to optimal action selection with respect to reward signals. POMDPs are not concerned with the issue of grounding their abstract representations in physical experience (see below). OACs provide more generic formalisations of actions in a cognitive system, also allowing for non-probabilistic representations in which action selection may not be the primary goal.

OACs also facilitate the learning of their associated prediction functions, an idea which is closely related to statistical structure learning [25, 26, 27, 28, 9, 8, 19], and learn how successful their executions are over particular time windows. In particular, in early development, when actions are likely to be unsuccessful, it is important to ensure that such execution uncertainties can be reasoned about. The storage of statistical data concerning execution reliability also has important applications to probabilistic planning [19], where an OAC's probability of success can be utilised to compute optimal plans. Consistently successful plans can then be memorised for future reference.

Affordances: OACs combine the representational and computational efficiency of STRIPS rules [7] and the object- and situation-oriented concept of affordance [29, 30]. Affordance is the relation between a situation, usually specified to include an object of a defined type, and the actions that it allows. While affordances have mostly been analysed in their purely perceptual aspect, the OAC concept defines them more generally as state-transition functions suited to prediction. Such functions can be used for efficiently learning the multiple representations needed by an embodied agent for symbolic planning, execution, and sensorimotor control.

Grounding and Situatedness: OACs reflect a growing consensus concerning the importance of grounding behaviour in sensorimotor experience, which has been stressed in the context of embodied cognition research (see, e.g., [11, 31, 32, 33]). To build a truly cognitive system, it is necessary to have the system's representations grounded by interacting with the physical

world in a closed perception-action loop [32]. OACs are necessarily grounded by their execution functions (Section 6), and are learnt from the sensorimotor experiences of the robot (Section 5). Thus, OACs realise grounding by "successful verification" [5] in an ongoing learning process.

The ability of OACs to formalise sensorimotor processes on different levels of the cognitive hierarchy allows high-level abstract actions to be formally grounded in sensory motor experience by means of lower-level actions. We have exemplified this by a "Birth of the Object" process [34, 35] described in Section 8.1. By this process, rich object descriptions and representations of grasping affordances (i.e., the association of potential grasping options to an object and their associated success likelihoods) emerge through interactions with the world. As we outline in Section 8.1, this process can be understood as the concatenation of several low-level perception-action interactions that are formulated in terms of OACs, leading to processes in which symbolic entities emerge (i.e., the notion of a specific object) and can be used on the planning level. Note that this is very much in line with prior work by others [6, 36] where representations and actions are likewise grounded through interaction. Differences in the specificities of our visual and motor representations compared to [6, 36] are discussed in detail in [35].

Modularity: The principle of modularity is widespread in cognitive process modelling (e.g., vision [37, 38] and motor control [39, 40, 41]), allowing the agent to make use of acquired perception and action competences in a flexible and efficient way. As we will demonstrate in Section 7, this concept is also inherent in the structure of OACs: OACs often operate at increasing levels of abstraction, each with a particular representation of situations and contexts. For instance, we will outline three examples of OACs for grasping objects. On the lowest level, continuous end-effector poses are associated to visual feature relations for grasping completely unknown objects. This OAC can be used to model reactive or affordance-based behaviours (see [42, 30]) as outlined in Section 7.2.2. At an intermediate level in another grasp-related OAC (described in Section 7.3), grasp densities are used to hypothesise possible grasps when the agent has some object knowledge [43]. Finally, at the highest level, plans effectively use grasps to manipulate objects on an abstracted symbolic scene representation (see Section 7.4).

Hybrid control schemes: OACs can be seen as a unifying representation for modelling control schemes in hybrid (i.e., discrete-continuous) dynamical systems (see, e.g., [44]). In this way, they are related to the idea of hybrid

control in systems which combine discrete events with continuous dynamics. In most practical cases, hybrid control architectures formalize discrete abstractions of inherently continuous control problems. For example, the task of manipulating an object can be decomposed into four subtasks: (1) reaching the grasp position, (2) grasping the object, (3) moving the object, and (4) placing the object at the goal position. From a hybrid control point of view, the subtasks associated with object manipulation can be described as discrete events, e.g., represented as finite state machines with continuous dynamics for each state. Each of the states might represent low-level, continuous controller operating on motor torques, sensor readings, etc., with discrete state transitions triggered by specific conditions of the lower-level controllers, or by external environmental stimuli.

Such hybrid control schemes can be implemented with OACs representing states, and their control programs implementing the low-level controllers. However, OACs can provide more than state models in hybrid dynamical systems. OACs can model open-loop, one-shot actions with stochastic outcomes, and be stacked into hierarchical architectures containing different layers of abstraction. At higher, symbolic levels, OACs can also be composed from other OACs, to be used for new tasks in different contexts.

Learning, Evaluation, and Memorisation: Cognitive agents must learn from past experience in order to improve their own development, a task that typically requires a form of memory as a means of tracking prior interactions (see, e.g., [45]). While memory itself is not often a problem, such processes must ensure efficient representation, with properties like associative completion and content addressability [46, 47, 48, 49], to enable machine learning from stored instances presented over a period of time.

Learning is also modularised through the OAC concept. In our example OACs, the lowest-level OAC learns the difference between successful and unsuccessful grasps. Using this as a base, another OAC learns alternative object-specific ways of posing the hand. Again, building on this OAC, another OAC learns the abstract preconditions and effects of grasping. Careful maintenance of the attribute spaces of the different OACs allows systems to benefit from the modularity of the information learnt for each OAC. As outlined in Section 8, the OAC formalism ensures that relevant data for learning is stored (in terms of "experiments"), and that learning is taking place at all times at all levels (even when learning is not the explicit goal of the agent).

4. Defining OACs

Our OAC definition is split into two parts, (1) a *symbolic description* consisting of a prediction function [DI-2] defined over an attribute space [DI-1], together with a measure of the reliability of the OAC [DI-6], and (2) an *execution specification* [DI-3] defining how the OAC is executed by the embodied system and how learning is realised [DI-5] by verification [DI-4].

This separation is intended to capture the difference between the knowledge needed for cause and effect reasoning (represented in the symbolic description), and the procedural knowledge required for execution (encapsulated in the execution specification). Since we do not constrain the form of the attribute space, OACs are not limited to continuous or discrete representations of actions. Instead, as we will see in Section 7, our definitions are flexible enough to accommodate both kinds of representations.

In the remainder of this section we will provide a formal definition of an OAC's symbolic description.

Definition 4.1. *We call the properties of the world captured by an OAC* ***attributes****. Each attribute has an associated range of possible values that can be assigned to that attribute.*

Intuitively, attributes can represent any sensed or derived property that we want our OACs to capture. In particular, Definition 4.1 does not make any commitments about attributes being continuous, discrete, or Boolean. This provides the OAC formalism with the flexibility to reason about very different problem spaces.

Definition 4.2. *An* ***attribute space*** $\mathcal{S}$ *is the set of all possible assignments of values to a set of attributes. A* ***state*** $s \in \mathcal{S}$ *denotes a (possibly partial) assignment of values to the attributes in the space.*

Since we have not limited the form of the attributes we permit, an attribute space can be very expressive, and an individual state description can abstract over a possibly large number of real-world states. Even a complete individual state in the OAC's attribute space can capture a possibly infinite number of real-world states. For example, a complete state specification that includes the assignment of the value "full" to the attribute "statusGripper" represents all the world states where the gripper is full, provided the other attributes of the world state are consistent with those of the OAC's state.

We also allow state descriptions to be partial, where values are only specified for a subset of the attributes in the space. For example, if the value "full" is assigned to the attribute "statusGripper", and no values are specified for any of the other attributes in the state space, then the resulting partial state denotes the set of all states where the gripper is full, regardless of the other attribute values. As a result, this state representation provides a powerful method for OACs to abstract over large state spaces.

We now turn our attention to formally defining OACs.

Definition 4.3. *An **Object-Action Complex (OAC)** is a triple*

$$(E, T, M) \tag{1}$$

where:

- *E is an identifier for an execution specification,*
- *$T : \mathcal{S} \rightarrow \mathcal{S}$ is a prediction function defined on an attribute space $\mathcal{S}$ encoding a model of how the world (and the agent) will change if the execution specification is executed, and*
- *M is a statistical measure representing the success of the OAC in a window over the past.*

Definition 1 characterises OACs using three main components. In the examples we will discuss here, the execution specification E identifies a single CP whose execution is modelled by the OAC. This means that multiple OACs can share the same underlying CP.[2]

In general, much of the actual world state will be irrelevant for most OACs. Therefore, we stipulate that the attribute space $\mathcal{S}$ captures all and only those attributes of the world that are needed for T to make its predictions. Thus, for a given OAC, $\mathcal{S}$ will often omit sizable portions of the sensed world, but may include specialised attributes derived from multiple sensors. Since observations are costly in real world systems, we can use the reduced space of $\mathcal{S}$ to constrain observations and allocate system resources more efficiently, resulting in a reduced sensor load for verifying OAC execution.

M codes an evaluation of the OAC's performance over a time window in the past. Given the diversity of attribute spaces we can define for OACs, M

[2]We will discuss more complex execution specifications in Section 6.

must be flexible enough to capture the reliability of many types of prediction functions. As a result, we allow each OAC to define M as a statistical measure appropriate for its needs. Thus, different OACs in a single system might define M in very different ways. For example:

- In a simple domain where an OAC is used until it fails and then is never used again, we might define M as a Boolean flag that indicates whether the OAC has failed.
- In a more complex domain where M tracks the accuracy of an OAC's prediction function over a certain time window in the past, we might define M as a pair made up of the expected value of the OAC's performance and the sample size used to compute the expected value.
- In even more complex domains it might be convenient to store statistical data beyond the expected value. For example, lower-level OACs might maintain statistical information about the differences between observed and expected changes in a number of specific attributes.

Note that the size of the temporal window over which M is collected is OAC dependent. In general, during learning (where large changes can significantly affect the success likelihood) smaller windows might be appropriate to judge whether learning is making good progress, whereas in the case of a mature OAC a larger window (and hence a more stable estimate of the success likelihood) might be appropriate.

To provide some intuition, we can imagine an agent with the following example OACs, defined on very different attribute spaces. These examples are described more formally in Section 7.

Ex-1 *An OAC that encodes how to push an object on a table based on the agent's end-effector pose space and the location of the object.* In this case, the OAC might predict the position of an object after a pushing action by the end-effector, depending on the velocity and force vector as well as the shape of the object. For M, the OAC might maintain the average deviation (over a certain time window) of the prediction of the position and the measured position after pushing the object.

Ex-2 *An OAC that encodes how to grasp an unknown "something" in a scene.* In this case, the OAC might predict the success or non–success (e.g., when the "something" is out of reach) of the grasping attempt. For M,

this OAC might store the likelihood of a successful grasp over a time window in the past.

Ex-3 *An OAC that encodes how to grasp a specific object in a specific scene suggesting an optimal gripper pose.* In this case, the OAC might also predict the success or non–success (e.g., in case the object is in a non-graspable pose) of the grasping attempt. For M, this OAC might store the likelihood of successfully grasping the object over a time window in the past.

Ex-4 *An OAC that encodes how to grasp an object for the purpose of planning (e.g., to systematically clean a table).* In contrast to Example Ex-3, at the planning level the precise control information required to grasp an object is not relevant. Rather, higher-level attributes such as the object affordances that become executable after a successful grasp need to be coded (e.g., the objects that are now movable to a shelf). For M, this OAC might store the likelihood that the grasp is successful over a time window in the past.

We will provide more detailed definitions of these example OACs and their reliability measures in Section 7. First, however, we will motivate the discussion of how OAC-based learning is formalised with the following definition.

Definition 4.4. *Let* `execute` *be a function with side effects that maps an OAC, defined on an attribute space $\mathcal{S}$, to a triple of states called an* ***experiment****, i.e.,*

$$\texttt{execute} : (E, T, M) \rightarrow (s_0, s_p, s_r), \tag{2}$$

where:

- $s_0 \in \mathcal{S}$ *is the state of the world before performing the OAC's execution specification,*
- $s_p \in \mathcal{S}$ *is the state of the world that T predicts will result from performing the OAC's execution specification in s_0, i.e., $s_p = T(s_0)$, and*
- $s_r \in \mathcal{S}$ *is the observed state resulting from actually performing E in state s_0.*

The side effect of this function is that the execution specification of the OAC is actually performed in the real world by the agent.

input : an OAC (E, T, M)
output: an experiment (`so`, `sp`, `sr`)
begin
    `so` = `stateCapture`(T);
    `sp` = T(`so`) ;
    `agentExec`(E);
    `sr` = `stateCapture`(T);
end

Algorithm 1: An implementation of `execute`.

Calling `execute` with an OAC causes the OAC's execution specification to be performed in the real world, producing an experiment as a result. This experiment is an *empirical event* dynamically created from the sensed and predicted states: its first element is derived by sensing the state of the world before execution, the middle term captures the OAC's prediction about the state that should have resulted, while the last element encodes the actual state of the world after execution. For example, an experiment for Example Ex-1 might include:

- the initial state of the end effector and the object,
- the predicted state of the object, and
- the actual state of the object after the execution.

We can imagine implementing `execute` with the pseudo-code in Algorithm 1. Here, `agentExec` is a function that causes the agent to perform the specified execution specification, and `stateCapture` is a function that captures the current state of the world, expressed in the attribute space of the given prediction function. For instance, in Example Ex-1, executing the "pushing OAC" launches a process that (1) captures the initial state, (2) invokes the prediction function on the initial state to predict the end state of the object after a pushing movement, (3) invokes the associated control program, (4) waits for it to terminate, (5) captures the resulting state of the object, and (6) reports all three states in the form of an experiment. We note that all OAC-specific processing takes place within the execution specification. For the rest of this paper, we will refer to the process of calling `execute` with a specific OAC as *executing an OAC*.

In our discussion up to this point, we have only considered a single OAC modelling a single control program, which simplifies the definition of the

execution specification: all we need to provide is the identifier of the control program that is to be executed. Given this mapping, `execute` has all the information it needs to invoke the specified control program, allow it to run until termination, and report the results as an experiment. In Section 7 we will see more detailed examples, and provide a discussion of how such one-to-one mappings can be built up in Section 8. However, we can also imagine much more complex specifications than the execution of a single control program. In particular, OACs might be defined in terms of other OACs, or sets of OACs. We will discuss this in more detail in Section 6.

As empirically grounded events, the experiments returned by `execute` can be used to update OACs in cycles of execution and learning (see Section 7) based on evaluations of their success [DI-4]. For instance, each of our example OACs might update their respective Ms on the basis of an experiment.[3] In the next section we explore particular learning problems in terms of OACs.

5. Learning OACs

The definition of an OAC as an entity that captures both symbolic and control knowledge for actions gives rise to a number of learning problems that must be considered for OACs to be effective. We note that each of these learning problems can be addressed by recognising that differences can exist between predicted states and actual sensed states. In practice, these problems may require different learning algorithms (e.g., Bayesian, neural network-like, parametric, non-parametric, etc.), and it is left to the OAC designer to choose an appropriate learning mechanism in each case.

As such, the following characterisations are intended to specify those aspects of the OAC that can be modified through learning, rather than a specific learning method. We consider four main learning problems, each of which is labelled in Figure 3, and illustrate these problems using the examples introduced in Section 4.

1. **Translation:** *(Learning the mapping of real-world states to OAC states)* This learning task produces the mapping from sensed world states to states in the OAC's attribute space. It also involves identifying and

[3] We leave open the possibility that an experiment might not be used immediately for learning, but could be stored in some type of short term memory (see, e.g., [45]) until resources for learning are available.

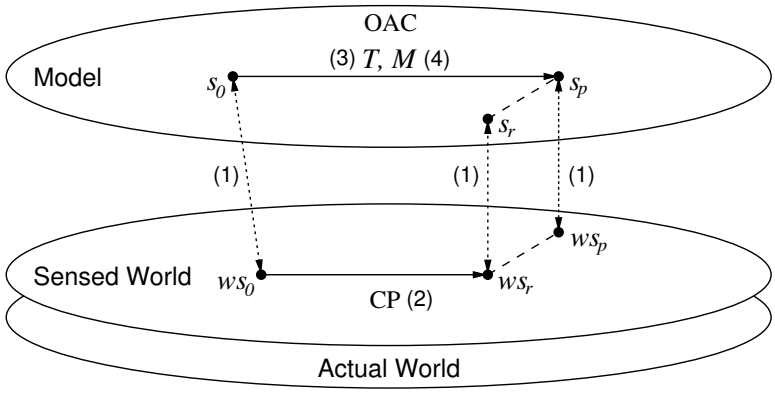


Figure 3: Graphical representation of the OAC learning problems: (1) Translation, (2) Control, (3) Prediction, and (4) Reliability.

adding to the OAC's attribute space those attributes of the world model that are required for effectively predicting interactions with the world. For instance, in Example Ex-1, this process would be responsible for adding new attributes (beyond object shape) such as the mass distribution of the object on the basis of more low-level sensory (visual and haptic) information, or the audio information caused by the object scraping along the surface.

2. **Control:** *(Learning control programs)* This learning task modifies an OAC's control program to minimise the distance between the world state ws_p predicted by the OAC and the actual sensed state ws_r. For instance, in Examples Ex-2 and Ex-3, when a grasp is not successful even though the OAC's T function predicts success, the control program can be modified to produce a successful grasp.
3. **Prediction:** *(Learning the prediction function)* This learning task modifies the prediction function to minimise the distance between a predicted model state s_p, and the actual resulting model state s_r. In Example Ex-1, this can be done by optimising the prediction function to produce a better estimate of the final state of the object after a push.
4. **Reliability:** *(Learning the prediction function's long term statistics)* This learning task updates the OAC's reliability measure M to reflect the long-term success of the OAC. In Examples Ex-2, Ex-3, and Ex-4, this process might record the last 100 attempts, and evaluate how many had been successful.

We reiterate that all of these learning problems can be addressed by recognising the differences between predicted states and actual sensed states as captured by experiments (i.e., through ongoing verification). However, the details and specifications of how each of these learning tasks might be performed at a given level of abstraction may vary wildly depending on the details of the attribute space. One of the critical contributions of this work is in enumerating and formalising these problems within the OAC framework.

In the following sections, we will use a set of common function names to denote each of these learning problems. Although these functions would have to be appropriately tailored to a particular OAC if we were to actually implement them, we will simply refer to them as: `updateModel`, `updateCP`, `updateT`, and `updateM`, respectively, and assume that each function takes an experiment as an argument.

6. Representational Congruency and Hierarchical Execution

Before we introduce hierarchical executions of OACs in Section 6.2 and Section 6.3, we begin by discussing a fundamental problem connected to OAC modelling, and a structural property for OACs that was earlier referred to as *representational congruency*.

6.1. Representational Congruency

When an OAC is executed, all the states returned to the OAC by an experiment are defined within the OAC's attribute space. This means that even in *mature* OACs (i.e., OACs that are well developed and are undergoing very little additional modification), it is possible for there to be states of the actual world that may not be predicted or (adequately) captured in the OAC's attribute space. In such OACs, there is no guarantee that its performance could be enhanced even by introducing additional attributes (e.g., by means of `updateModel`, where we actually extend an OAC's attribute space). This is even more true for less mature OACs that do not have fully developed attribute spaces: OACs that are "missing" attributes may fail to make accurate predictions. As a result, OACs are only as effective at predicting the outcomes of interactions as their learnt models allow them to be.

Representational congruency is a property that aligns an OAC's attribute space with that of a control program (or another OAC, as we'll see in Section 6.2), ensuring the completeness of the OAC's prediction function is improved. To formalise this idea, we provide the following definition.

Definition 6.1. *Let $\boldsymbol{A} = (E, T, M)$ be an OAC defined on an attribute space $\mathcal{S}_A$, and let $\mathcal{S}_{sense}$ be the agent's "foundational attribute space" defined by the agent's set of sensors and the complete set of their possible values. $\boldsymbol{A}$ is said to be* ***representationally congruent*** *to the control program captured by E iff $\forall ws_0, ws_r \in \mathcal{S}_{sense}$ and $\forall s_0, s_p \in \mathcal{S}_A$, such that s_0 and s_p are the respective projections of ws_0 and ws_r into $\mathcal{S}_A$, and the execution of E by $\boldsymbol{A}$ in a sensed world state ws_0 gives rise to a sensed world state ws_r, then it follows that T maps s_0 to s_p.*

Note that representational congruency is not a necessary property of an OAC. Since our OAC definition doesn't say anything substantive about the prediction function, any function is permitted. However, prediction functions that consistently fail to produce sound and complete mappings (with respect to the actual sensed world) won't be useful for reasoning, even if they are permitted by the OAC definition. As such, representational congruency provides the logical underpinning for an OAC's attribute space and prediction function to accurately model real-world interactions.

As a result, representational congruency as described in Definition 6.1 is not a property that we assume OACs begin with. Instead it is a "target" property that OACs converge towards as they improve their underlying models. In this view, Definition 6.1 captures a types of completeness property that may not be fully achieveable in practice. However, the intuition behind this definition, that representationally congruent OACs correctly predict the states that result from the execution of a control program, is a property that is essential if OACs are to be effective at certain reasoning tasks.

In the next section we will discuss more complex configurations of OACs and what executing such OACs means to representational congruency and our notion of an experiment.

6.2. Towers of OACs

It is worth recognising that, beyond being attached to external sensors, there is no significant difference between the attribute space of an OAC and the sensed world within which a CP operates. A CP moves the agent from one state of the sensed world to another, while the execution of an OAC moves the agent from one state of its attribute space to another. Building on this correspondence, we can consider OACs that use the attribute space of another, more basic OAC, as their "sensed world" and define their execution specification in terms of these more basic OACs.

Generalising this idea results in "towers" of OACs where each OAC stands in one-to-one relation with an OAC (or a control program in the base case) that is beneath it in the tower. In such cases, the execution specification of each OAC is just the recursive invocation of the OAC beneath it in the tower. Calling `execute` for the highest-level OAC results in a stack of calls to `execute`, one for each level of the tower, where each OAC invokes the OAC at the next level down until the process terminates with the execution of a single control program. The experiment that results from this execution must then be returned back up the tower, and appropriately translated into the attribute space of each OAC, as the result of each `execute` call.

For instance, consider the planning-level grasping OAC in Example Ex-4, operating in a discrete state space with an abstract description of objects and their graspability. This OAC's execution specification could invoke the OAC in Example Ex-3 which operates in the lower, continuous space of concrete gripper poses. A call to the high-level OAC in Ex-4 would then result in a call to the lower-level OAC in Ex-3 which computes a concrete end effector pose and triggers the execution of the control program. At each level, the resulting experiments would be passed back to the respective OACs.

We can modify our definition of representational congruency to permit towers of OACs. Recall that Definition 6.1 required an attribute space derived from the agent's sensor set. To extend representational congruency, we alter this definition to refer to the attribute space of the execution function in general. This results in the following revised version of Definition 6.1:

Definition 6.2. *Let $\boldsymbol{A} = (E, T, M)$ be an OAC defined on an attribute space $\mathcal{S}_A$, and let $\mathcal{S}_E$ be the attribute space of the OAC or CP specified by E. $\boldsymbol{A}$ is said to be **representationally congruent** to the execution specification captured by E iff $\forall s'_0, s'_p \in \mathcal{S}_E$ and $\forall s_0, s_p \in \mathcal{S}_A$, such that s_0 and s_p are the respective projections of s'_0 and s'_p into $\mathcal{S}_A$, and the execution of E by $\boldsymbol{A}$ in a state s'_0 results in a state s'_p, then it follows that T maps s_0 to s_p.*

We note that while Definition 6.2 is sufficient for describing representational congruency in towers of OACs, it will need further extension if we are to capture OACs with even more complex execution specifications, since the attribute spaces for such constructs could be substantially more complex.

We have also discussed how an experiment resulting from executing OACs in a tower must be passed back to each constituent OAC, and translated into the attribute space of that OAC. This means that the attributes and values

of a higher-level OAC's attribute space must be definable in terms of the attributes and values of the lower-level OAC. To ensure this property holds, we impose the following restriction on the attribute spaces of towers of OACs.

Definition 6.3. *Let* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *be OACs and let* $\mathcal{S}_A$ *and* $\mathcal{S}_B$ *be the attribute spaces of* $\boldsymbol{A}$ *and* $\boldsymbol{B}$*, respectively. If* $\boldsymbol{A}$ *has an execution specification defined in terms of* $\boldsymbol{B}$*, then all the attributes of* $\mathcal{S}_A$ *must be derivable from the attributes of* $\mathcal{S}_B$*. In such cases we will say that* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *are* ***hierarchically defined****.*

We will see examples of towers of OACs in Section 7.4 and Section 8.2.

6.3. One-to-Many Execution

One-to-one mappings are not the only kind of relationship we can envision for OACs. We can also imagine more complex scenarios, where an OAC is mapped to a sequence of OACs or control programs, or has an execution specification that involves iteration, conditional invocation, or parallel execution. For example, an OAC for opening a door might be comprised of a sequence of lower-level OACs that include actions to approach the door, grasp the doorknob, twist the doorknob, pull on the doorknob, etc. In order to execute such a higher-level OAC, each of these lower-level OACs must be successfully executed in the correct sequence.

A formal definition that permits one-to-many execution specification requires ordering constraints and success criteria for each of the sub-OACs. Furthermore, a correct understanding of the execution specification for such OACs must, like the one-to-one case, rest on recursively calling the `execute` function and continually monitoring the execution of the underlying OACs. We will not provide a detailed definition of such complex execution behaviour in this paper. Instead, we leave the specification and learning of such behaviours as an area for future work.

7. Examples of OACs

In this section, we give formal descriptions for a number of OACs. Some of these OACs have already been discussed informally as part of our running examples (Ex-1—Ex-4), while others are new. For each OAC, we provide a definition of its attribute space ($\mathcal{S}$), prediction function (T), success measure (M), and execution specification (E). We also discuss learning in these OACs, and show how they can be embedded within procedural structures

Section	Name	Attribute space/T	M	Learning
7.1 (Ex-1)	**AgnoPush**	End effector's pose space, object location and shape	Average deviation of prediction from actual final position	T, M
7.2 (Ex-2)	**AgnoGrasp**	Space of coplanar contour pairs, gripper status	Long term probability of successful grasp	CP, M
7.3 (Ex-3)	**ObjGrasp**	Object model, gripper status	Long term probability of successful grasp	CP, M
7.4 (Ex-4)	**PlanGrasp**	Logic-based rules	Long term probability of correct result prediction	T, M
7.4 (Ex-4)	**PlanPush**	Logic-based rules	Long term probability of correct result prediction	T, M

Table 1: Summary overview of example OACs.

to produce more complex behaviour. In Section 8, we will present examples of these OACs interacting with each other, to demonstrate grounding and planning. Table 1 provides an overview of the example OACs for comparison.

*7.1. Example Ex-1: Object Pushing (**AgnoPush**)*

In this example we define an OAC **AgnoPush** which models a pushing action that moves objects in a desired direction on a planar surface without grasping. Pushing as a nonprehensile action cannot be realised with sufficient accuracy to ensure a given object can be moved to a desired target in one step, i.e., by applying one pushing movement. If a higher-level planner specifies that object o should be pushed to a certain target, **AgnoPush** needs to be applied iteratively in a feedback loop until the target location is eventually reached. To achieve this, the system needs to know how objects move when short pushing actions are applied to them. In particular, the motion of the pushed object depends on various properties including shape, mass distribution, and friction. Here we will focus on shape. (A more detailed

description can be found in [50].)

7.1.1. Definition of ***AgnoPush***

Defining $\mathcal{S}$: Some prior knowledge needs to be available before **AgnoPush** can be learnt. In particular, we assume that the robot knows how to move the pusher (e.g., the robot hand or a tool held in its hand) along a straight line in Cartesian space. We also assume that the robot knows how to localise the observed objects by vision. The central issue for **AgnoPush** is to learn to predict the object's movement in response to the pusher's movement. To this end, the robot needs information about the object's shape, its current location on the planar surface, the duration of the pushing movement, and its direction relative to the point of contact on the object's boundary. We represent the object's shape by a 2D binarized image, such as those shown in Figure 4. Such images are sufficient as shape models (as opposed to full 3D shape models) because **AgnoPush** only encodes the response to an applied pushing action for objects that do not roll on planar surfaces.

More formally, $T_{\mathbf{AgnoPush}}$ is defined on the attribute space

$$\mathcal{S} = \{\text{bin}(o), \text{loc}(o), \tau, a\},$$

where $\text{bin}(o)$ is the shape model in the form of a binary image of the object to be pushed, $\text{loc}(o)$ denotes the initial location of the object o, τ is the duration of the push, and a denotes the parameters describing the pushing movement, i.e., the contact of push on the object's boundary and the direction of the movement of the pusher.

Defining T: Based on the information in this attribute space, we can predict the object's new location using the transformation

$$T(\text{bin}(o), \text{loc}(o), \tau, a) = V(\text{bin}(o), \text{loc}(o), a)\tau + \text{loc}(o), \tag{3}$$

where V is the function predicting the outcome of the push in terms of the object's linear and angular velocity.

T returns the expected position and orientation of the object after it has been pushed at a given point of contact and angle with constant velocity for a certain amount of time. The angle of push is defined with respect to the boundary tangent. These parameters are fully determined by the object's binary image and the pusher's Cartesian motion. Thus,

$$T : \mathcal{S} \longrightarrow \{loc(o)\}$$

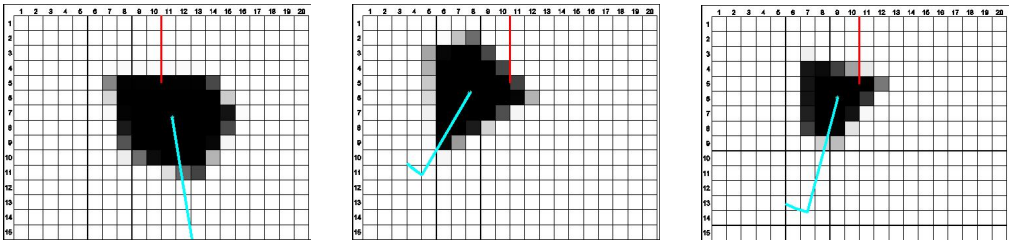


Figure 4: Samples of low resolution object images used as input to the neural network.

maps an initial state $(\text{bin}(o), \text{loc}_o(o), \tau, a)$ containing a concrete shape $\text{bin}(o)$, a location $\text{loc}_o(o)$ before the action and a specific poking action parameterized by (τ, a) to a predicted location $\{\text{loc}_p(o)\}$ after the action.[4]

Defining M: The statistical evaluation M measures how close the predicted object movement is to the real object movement over a certain time window. We define a metric $\text{d}(\text{loc}_p(o), \text{loc}_r(o))$ to measure the difference between the expected and actual object movement on the planar surface. The expectation of **AgnoPush**'s performance after N experiments is thus given by

$$M = \frac{1}{N} \sum_{i=1}^{N} \text{d}(\text{loc}_p(o)_i, \text{loc}_r(o)_i),$$

where i denotes different pushing trials (see Figure 5, right).

Defining E: An impulse to push an object in a certain direction must be provided by a higher-level cognitive process. The appropriate parameters to the pushing control program can be determined based on the available prediction function T. These issues will be discussed in Section 7.1.2. However, the control program modelled by **AgnoPush** is neither object nor target dependent. This means that the execution specification of this OAC simply calls the pushing control program with parameters a and τ computed by an external process.

Calling `execute` results in an experiment of the form

$$(\{\text{bin}(o), \text{loc}_o(o), \tau, a\}, T(\text{bin}(o), \text{loc}_o(o), \tau, a), \text{loc}_r(o))$$

[4]For brevity in Section 7 and Section 8 , we will often provide partial state descriptions when discussing T and the experiments resulting from `execute`, highlighting the significant parts of the state, rather than simply reporting complete states.

that is created by performing four major functions:

1. Capturing the initial state: For **AgnoPush**, this requires both extracting the binary image of the object $\text{bin}(o)$ and its location $\text{loc}_o(o)$, and acquiring the pushing movement parameters a.
2. Capturing the predicted resulting state: This is done by calculating $T(\text{bin}(o), \text{loc}_o(o), \tau, a)$.
3. Executing the execution specification: The pushing movement is performed by calling the pushing control program with parameters a and τ.
4. Capturing the actual resulting state: This is done by localising the object after the push, i.e., by observing $\text{loc}_r(o)$.

When the task is to push an object towards a given target location, the robot can solve this by successively applying `execute` in a feedback loop until the goal has been reached. Note that in this example the control program that realises straight-line motion of the pusher in Cartesian space is fixed and does not need to change while learning **AgnoPush**.

*7.1.2. Learning in **AgnoPush***

Learning in **AgnoPush** affects both its prediction function and its long-term statistics. A process for learning the prediction function (denoted by the function `updateT`) is realised using a feedforward neural network with backpropagation. The trained network encodes a forward model for object movements that have been recorded with each pushing action. To ensure that **AgnoPush** can be applied to different objects, the shape is specified in the form of a low resolution binary image, which is used as input to the neural network. Function T is updated incrementally based on the observed movements of the pushed objects. Statistical evaluation is also done incrementally as experiments are performed (a process denoted by the function `updateM`). Note, however, that since the prediction function T changes during learning, the statistical evaluation only converges to the true accuracy of the behaviour once T becomes stable (see Figure 5).

There are two modes of operation in which we consider **AgnoPush**:

A. Initial learning of the prediction function T, where the pushing movements encoded by the parameter a are randomly selected, and
B. Pushing the object towards a given target, where the current pusher movement a is determined based on the previously learnt prediction function and the given target location.

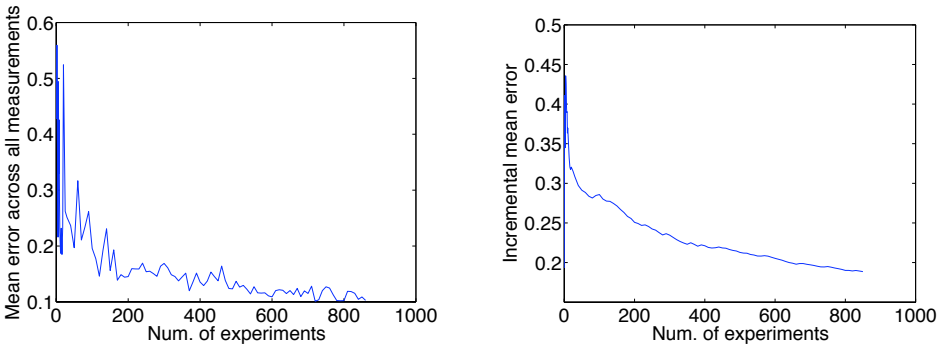


Figure 5: Mean error of robot pushing. The left figure shows the mean error of the predictor on the available data, i.e., after each update of the predictor we evaluate its performance on all previous experiments. The right figure shows the incremental statistical evaluation as realised by `updateM`. Four different objects were used in the experiment.

Figure 6: Pushing behaviour realised by **AgnoPush** after learning prediction function T.

As described above, the prediction function T is given in Equation (3), where velocity V is encoded by a neural network with the binary image of an object, the point of contact and the direction of the pusher movement used as input values, and the predicted final position and orientation of the pushed object as output. In mode B, we calculate the optimal pusher movement a (i.e., the point of contact and the direction of the push) by first extracting the object's binary image and determining the desired Cartesian movement of o from its current location towards the target location. The neural network is then inverted using nonlinear optimisation (see [50] for details). The resulting behaviour is shown in Figure 6.

The learning process has been implemented using explorative behaviour as shown in Algorithm 2. In this context, `updateT` estimates the weights of the neural network (for details on the learning algorithm, see [50]). To ensure that the data used for training is not used to estimate the performance of

```
while true do
    a = SelectRandomMotion; bin(o); loc_o(o);
    expr = execute(AgnoPush);
    if d(loc_o(o), loc_r(o)) > ε then
        updateM(expr);
        updateT(expr);
    end
end
```

Algorithm 2: Explorative behaviour to learn **AgnoPush**. The constant $\epsilon > 0$ is used to determine whether the object has moved or not.

the prediction function, `updateM` is always applied to the data before it has been used to refine the prediction function. This loop also demonstrates how OACs can be embedded in procedural structures. We will see more examples of such procedures in the following sections.

*7.2. Example Ex-2: Object Independent Grasping (**AgnoGrasp**)*

Next, we consider an OAC **AgnoGrasp** that predicts the success of attempts to grasp unknown objects, based on an associated grasping hypothesis (specified in terms of the 6D pose of the gripper) for a co-planar contour pair (see Figure 7(a),(b) and [51]). Such grasp hypotheses are "agnostic" to the object being grasped, hence the name of the OAC. As a result, **AgnoGrasp** represents a visual feature/grasp association that enables an unknown "something" to be grasped (see Figure 7(d)).

*7.2.1. Definition of **AgnoGrasp***

Defining $\mathcal{S}$: We note that two co-planar contours define a plane which determines the orientation normal of the pose (i.e., two orientation parameters) for any possible grasp. The position and main orientation of a contour in 3D space determines the position of the 6D pose and the one remaining orientation parameter of the grasping hypothesis. This allows us to associate a grasp hypothesis $G^{\mathrm{H}}(C_i, C_j)$ with any pair of co-planar contours (C_i, C_j) (see Figure 7(b)). Such grasp hypotheses can then be executed by the system.

Formally, **AgnoGrasp** is defined on the attribute space:

$$\mathcal{S} = \{\texttt{statusGripper}, \Omega, \texttt{statusGrasp}\}\,.$$

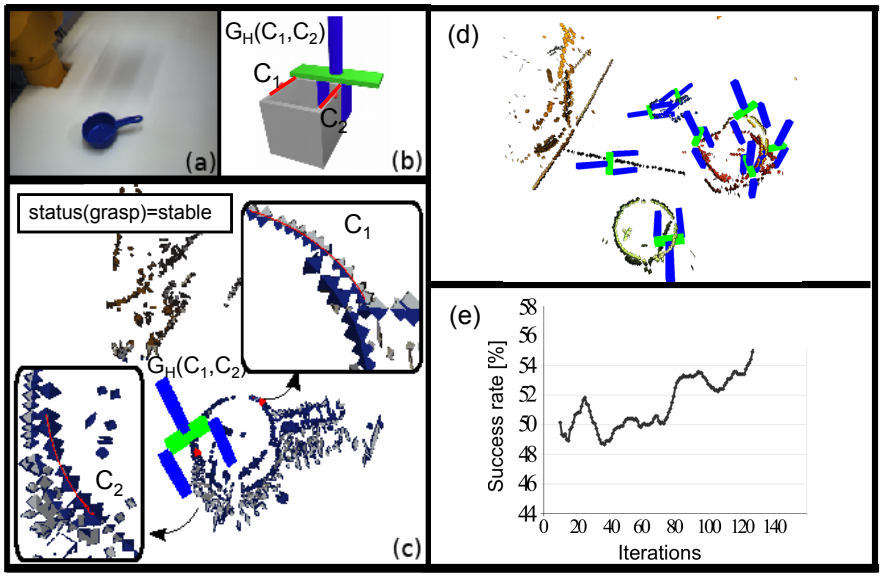


Figure 7: (a) The image of the scene captured by the left camera. (b) A possible grasping action type defined by using the two coplanar contours C_1 and C_2 shown in red. (c) A successful grasping hypothesis. The 3D contours from which the grasp was calculated are shown. Note that the information displayed is the core of an experiment. (d) A selected set of grasping hypotheses generated for a similar scene. (e) Change of performance as measured by M as a result of the learning process.

This attribute space contains the set Ω containing the co-planar contours in the scene and the status of the gripper `statusGripper` which either can take the value 'full' or 'empty'. In particular, it requires that (1) there are co-planar contours $C_i, C_j \in \mathcal{C}$ in the scene (i.e., the set of co-planar contours Ω is not empty), and (2) the gripper is empty.

Defining T: AgnoGrasp's prediction function determines the value of the attribute `statusGrasp`:[5]

$$\texttt{statusGrasp} \in \left\{ \begin{array}{l} \texttt{undefined}, \texttt{noplan}, \texttt{collision}, \\ \texttt{void}, \texttt{unstable}, \texttt{stable} \end{array} \right\}.$$

[5]Note that the current implementation of our learning algorithm only uses two classes, *success* which is equivalent to `stable`, and *failure* which corresponds to all other states except `noplan`, where the generated experiments are ignored for learning. More advanced learning algorithms might also use extended information on the stability of a grasp.

The possible values of `statusGrasp` each capture an outcome of the execution of **AgnoGrasp**. Before execution, `statusGrasp` is set to `undefined`. After selecting a specific grasping hypothesis, a motion planner tries to find a collision-free path that allows the arm to reach the pregrasping pose associated with the grasping hypothesis, which may result in a number of possible outcomes. If the planner fails to find a suitable trajectory or decides there is none, execution stops, and the result is `noplan`. If the hand unexpectedly enters into a collision, execution stops at that point, and the result is `collision`. If the closed gripper is determined to be empty, the result is `void`. If the gripper closes further while lifting the object, the result is `unstable`. Otherwise, the grasp is deemed successful, and the result is `stable`. In our case, $T_{\mathbf{AgnoGrasp}}$ simply maps to a state where $\texttt{statusGrasp} = \texttt{stable}$ holds.[6]

Defining M: In **AgnoGrasp**, the reliability measure $M_{\mathbf{AgnoGrasp}}$ is simply defined as the percentage of successful grasps in a time window of 100 grasping attempts.

Defining E: Like **AgnoPush**, **AgnoGrasp**'s execution specification is based on executing a low-level control program. In the case of **AgnoGrasp**, the CP requires as input a pair of co-planar contours from the scene (where a grasp hypothesis can be computed) that is chosen from the set of contours Ω. Thus, prior to execution, many grasping hypotheses from co-planar contour pairs are computed and a single pair is chosen for execution.[7]

This means that when performing `execute`, the initial state is given by:

$$\{\Omega, \texttt{statusGripper}\},$$

where Ω is the set of contours. The predicted state is simply an assertion that $\texttt{statusGrasp} = \texttt{stable}$ holds. After the chosen grasp hypothesis is performed, the grasp status $\texttt{statusGrasp}_{t+1}$ is sensed. This results in an experiment of the form:

$$(s_0, \texttt{statusGrasp}_{t+1} = \texttt{stable}, \texttt{statusGrasp}_{t+1}).$$

[6]The use of a constant mapping here not only represents the most likely outcome but, for certain reasoning tasks, the most wanted outcome. Space prohibits a comprehensive discussion of the motivation behind such mappings.

[7]In practice, the pair is chosen according to a ranking criterion. See [51] for details.

```
while true do
    compute contours pairs and associated grasping hypotheses
    expr = execute(AgnoGrasp);
    updateCP(expr);
    updateM(expr);
    drop object
end
```

Algorithm 3: A simple learning cycle for **AgnoGrasp**.

(See Figure 7(c) for the main components of an experiment.) Each experiment can either be used directly for on-line learning, as in the learning cycle in algorithm 3, or stored in an episodic memory for off-line learning at a later stage (see [52] for details).

*7.2.2. Learning in **AgnoGrasp***

In **AgnoGrasp**, learning affects the execution of the control program (through the `updateCP` function), and the updating of long-term statistics (via `updateM`; see Figure 7(e)). We do not consider other learning problems and, in particular, the OAC's prediction function always remains constant. Learning modifies the selection of the most promising grasping hypothesis and, thus, the control program underlying the execution function. In practice, the optimal choice of grasps depends on certain parameters, such as contour distance and the object position in working space (see Figure 7(d)). Based on an RBF network (see [52] for details), a function estimates the success likelihood that a certain grasp has been learnt in a cycle of experimentation and learning.[8] Algorithm 3 formalises this exploratory behaviour, which realises a simple learning cycle for **AgnoGrasp**.

*7.3. Example Ex-3: Object Specific Grasping (**ObjGrasp**)*

In this example, we consider an OAC **ObjGrasp** that models the grasping options for a specific object, and their associated success likelihoods, by means of *grasp densities* (see Figure 8 and [43]).

[8] In practice, such learning has provided an increase in the success rate from 42% to 51% (see [52] for details). Note that since **AgnoGrasp** uses very little prior knowledge, a high performance cannot be expected except in trivial scenarios.

*7.3.1. Definition of **ObjGrasp***

Defining $\mathcal{S}$: Object models o_i are stored in an object memory $\mathcal{M}^{\mathrm{O}}$.[9] An object model includes a learnt, structural object model that represents geometric relations between 3D visual patches (i.e., *early cognitive vision* (ECV) features [53]) as Markov networks [54]. In addition, it contains a continuous representation of object-relative gripper poses that lead to successful grasps by means of grasp densities [43]. Object detection, pose estimation, and the determination of useful gripper poses for grasping the object are all done simultaneously using probabilistic inference within the Markov network, given a scene reconstruction in terms of ECV features.

The attribute space for **ObjGrasp** is defined by

$$\mathcal{S} = \{\texttt{statusGripper}, \texttt{targetObj} = o, \texttt{statusGrasp}\}.$$

Here, the state description includes an attribute `targetObj` that specifies an object model o that is provided by the `execute` function as an input to the control program this OAC models. As notation, we will add a subscript to the OAC's name to identify this object model (e.g., $\textbf{ObjGrasp}_{\mathrm{Basket}}$). Like the two previous OACs, this model is chosen by processes external to **ObjGrasp**. The state description also includes `statusGripper` and `statusGrasp` as in Section 7.2, however, `statusGrasp` is only relevant to the predicted state.

Defining T: As with **AgnoGrasp**, the prediction function T always returns an assertion that `statusGrasp` = `stable` is true.

Defining M: The reliability measure M for **ObjGrasp** is defined as the cumulative outcome of statistics from executing this OAC (which is updated as part of a learning cycle; see Figure 9).

Defining E: Like **AgnoGrasp**, the execution of **ObjGrasp** requires its input parameters to be passed to a control program for execution. In the case of **ObjGrasp**, this parameter is the object to be grasped. When the `execute` function is performed, the process of capturing the initial state must:

1. access or reconstruct the current scene in terms of ECV features, and

[9] See Section 8.1 for more information about learning such models.

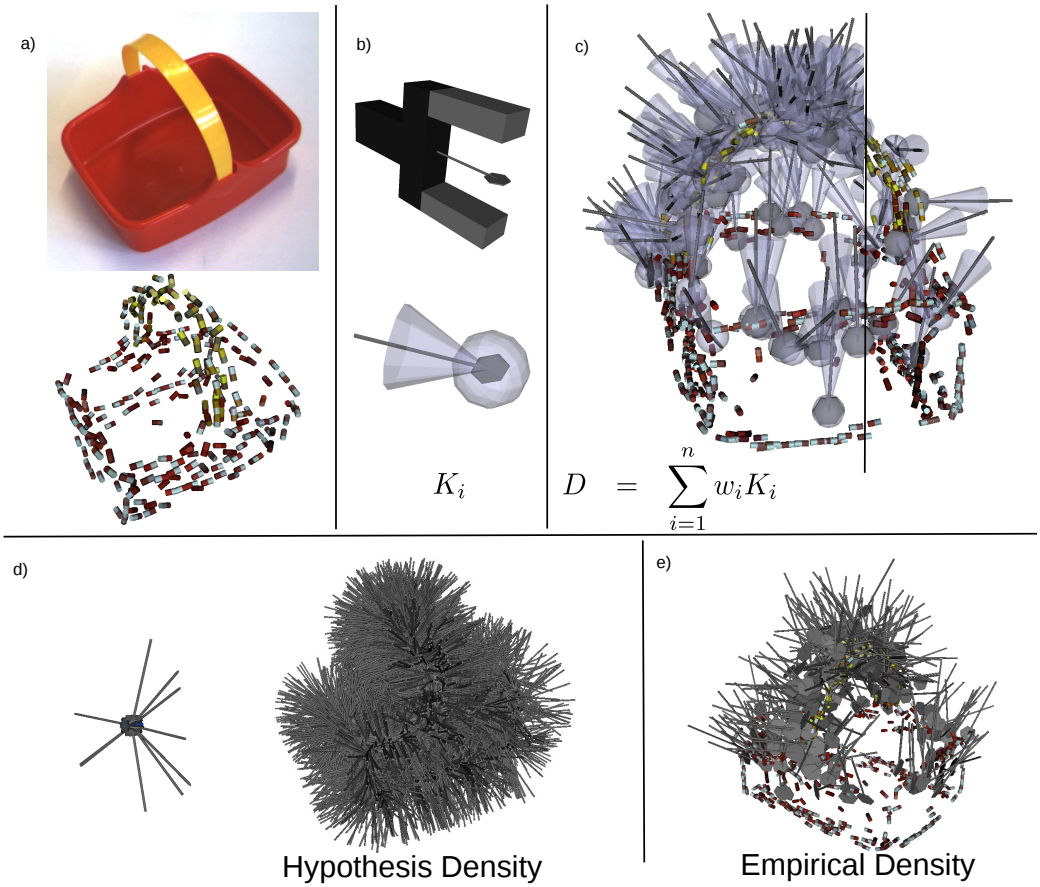


Figure 8: Mechanisms used by **ObjGrasp**. a) Objects (top) are represented as Markov networks [54] in terms of 3D ECV features [53] (bottom). b) In the following subfigures, gripper poses (grasps) are visualised as "paddles" (top). Grasp densities are obtained from individual grasps by kernel density estimation using $SE(3)$ kernels, as illustrated by unit-variance isosurfaces for 2 rotational and 3 positional degrees of freedom (bottom). c) A grasp density D associated with the basket (a). The right-hand side shows sparser samples for better visibility. d) Grasp hypothesis densities for specific objects such as the basket (right) are generated at uniform orientations around 3D ECV features (left). e) Empirical grasp density learnt by testing grasps drawn from a hypothesis density [43].

```
while true do
    compute ECV features
    expr = execute(ObjGrasp);
    updateCP(expr);
    updateM(expr);
    drop object
end
```

Algorithm 4: Exploration learning procedure for **ObjGrasp**.

2. retrieve the object model o from $\mathcal{M}^{\mathrm{O}}$, use it to locate the object, and determine a gripper position from the associated grasp density (see Figure 8).

Like **AgnoGrasp**, the OAC's prediction function returns `statusGrasp` = `stable`. The actual execution specification of the OAC encompasses a small, two-step control program:

1. First, a path planner generates a plan for manoeuvring the gripper to the intended position.
2. If such a plan is found, the CP executes the computed trajectory, and closes the gripper to grasp the object.

This yields a new state characterised by an attribute `statusGrasp` that can take on any of the values in the attribute space of the OAC **AgnoGrasp**, or the value `nopose`, which represents the case that no object instance can be reliably located. As a result of E, an experiment of the form

$$(\{\texttt{statusGripper}, o, \texttt{statusGrasp}\}, \texttt{statusGrasp}_{t+1} = \texttt{stable}, \texttt{statusGrasp}_{t+1}).$$

is returned.

We note that objects are always located within the currently-sensed part of a scene. Thus, it is up to other parts of the system to make sure that the scene reconstruction available to `execute` contains one and only one instance of the object o, e.g., by directing sensors accordingly.

*7.3.2. Learning in **ObjGrasp***

Algorithm 4 outlines how a higher-level process might acquire and refine grasping skills on a variety of objects. In this scenario, the scene contains up

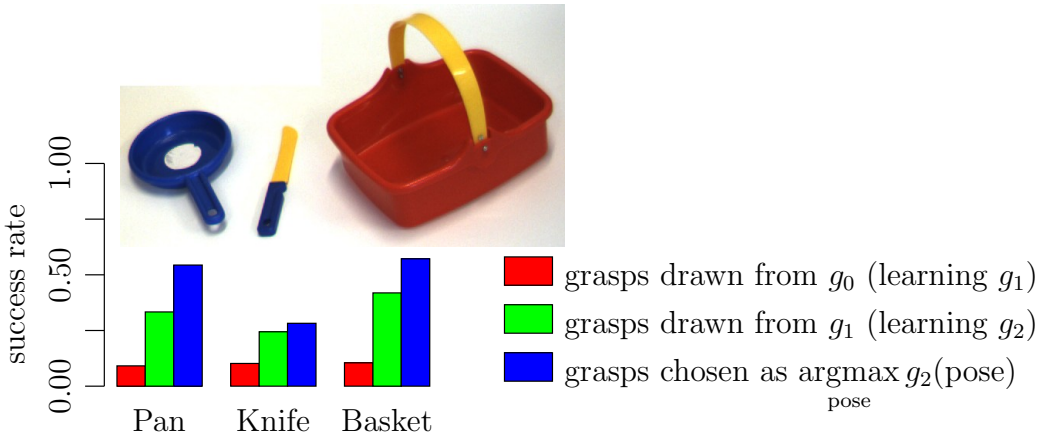


Figure 9: Evolving statistics M of `statusGrasp` = `stable` for the OACs $\mathbf{ObjGrasp}_{\text{Pan}}$, $\mathbf{ObjGrasp}_{\text{Knife}}$, and $\mathbf{ObjGrasp}_{\text{Basket}}$ over successive rounds of grasping trials [43]. In the first round, grasps are drawn from a hypothesis density g_0 generated from ECV features (Figure 8) for each object; the red bars show the empirical success rates, and the grasp density computed over the successful grasps is denoted g_1. In the second round (green bars), grasps are drawn from g_1, resulting in g_2. In the third round (blue), grasps are chosen as the maximum of g_2 for each object.

to one instance of each object of interest. The robot "plays" with the object by repeatedly grasping and dropping the object. This leads to a learning cycle similar to Algorithm 3, in which the system generates knowledge about the grasp affordances associated to the object.

*7.4. Example Ex-4: OACs for Planning (**PlanGrasp**, **PlanPush**)*

As a final example, we consider two high-level OACs suitable for planning: **PlanGrasp**, an OAC for grasping an object from a table, and **PlanPush**, an OAC for pushing an object into the reachable space so that it can be grasped [55]. Both of these OACs operate on discrete, attribute spaces defined in terms of a set of logical predicate and function symbols that denote properties and objects in the world. Such representations are standard in AI planning systems and we will structure our OACs in such a way that we can use prior planning work for building and executing plans.

*7.4.1. Definition of **PlanGrasp** and **PlanPush***

Table 2 shows a complete set of attributes that formalise a simple problem domain for picking up objects from a table and putting them onto a shelf. These attributes should be thought of as logical symbols (e.g., `clear`)

Attribute	Description
`clear(X)`	A predicate indicating that no object is stacked on `X`.
`focusOfAttn(X)`	A predicate indicating that object `X` is the focus of attention.
`gripperEmpty`	A predicate indicating that the robot's gripper is empty.
`inGripper(X)`	A predicate indicating that object `X` is in the gripper.
`onShelf(X)`	A predicate indicating that object `X` is on the shelf.
`onTable(X)`	A predicate indicating that object `X` is on the table.
`pushable(X)`	A predicate indicating that object `X` is pushable by the robot.
`reachable(X)`	A predicate indicating that object `X` is reachable for grasping by the gripper.

Table 2: A set of logical attributes for a simple planning domain.

with arguments represented by variables (e.g., `X`). Each ground term (e.g., $\texttt{clear(Obj}_0\texttt{)}$) has an associated truth value that is interpreted relative to the current world state.

Defining $\mathcal{S}$: To define the attribute spaces for **PlanGrasp** and **PlanPush** we restrict the set of attributes shown in Table 2. We define the attribute space for **PlanGrasp** in terms of the set of logical attributes:

$$\mathcal{S} = \left\{ \begin{array}{l} \texttt{focusOfAttn(X)}, \texttt{inGripper(X)}, \texttt{reachable(X)}, \\ \texttt{clear(X)}, \texttt{gripperEmpty}, \texttt{onTable(X)} \end{array} \right\}.$$

We also define the attribute space for **PlanPush** in terms of the attributes:

$$\mathcal{S} = \left\{ \begin{array}{l} \texttt{focusOfAttn(X)}, \texttt{reachable(X)}, \texttt{pushable(X)}, \\ \texttt{clear(X)}, \texttt{gripperEmpty}, \texttt{onTable(X)} \end{array} \right\}.$$

We note the only significant difference between these two attribute spaces is the inclusion of `inGripper(X)` for **PlanGrasp**, but not for **PlanPush**.

We also note that the representations used here could be more expressive (e.g., the arguments could be restricted to only allow objects of a particular type, or a multivalued logic could be used). In the interest of clarity we have used a simple representation: identifying the attribute space of an OAC is a challenging task no matter the level of abstraction, and learning the set of logical attributes in this OAC is a difficult process. (We recognise this as an instance of the "translation" learning problem.) A complete treatment of how this attribute space could be learnt is outside the scope of this paper.

Defining T: Given these attribute spaces, we can define T for each OAC as a pairing of initial conditions with predicted state descriptions (see Table 3). To

Name	Initial Conditions	Prediction
PlanGrasp	`focusOfAttn(X)`	`inGripper(X)`
	`reachable(X)`	`not(gripperEmpty)`
	`clear(X)`	`not(onTable(X))`
	`gripperEmpty`	
	`onTable(X)`	
PlanPush	`focusOfAttn(X)`	`reachable(X)`
	`not(reachable(X))`	
	`pushable(X)`	
	`clear(X)`	
	`gripperEmpty`	
	`onTable(X)`	

Table 3: Prediction functions T for planning-level grasping and pushing OACs.

specify the prediction function, both the initial conditions and the predictions are assumed to be conjunctions of specific attributes, i.e., all of the initial conditions must be true in the world for the prediction function to be defined, and all of the predictions are expected to be true in any state that results from the execution of the OAC. In terms of **PlanGrasp**, this means that if an object is the focus of attention, on the table, clear, reachable, and the agent's gripper is empty, then after executing this OAC we predict the object will be in the gripper, not on the table, and the gripper will no longer be empty. Likewise, for **PlanPush**, if an object is the focus of attention, unreachable, pushable, clear, on the table, and the agent's gripper is empty, then after executing this OAC we predict the object will be reachable.

Note that, like the other OACs we have discussed, these prediction functions do not make predictions in all states. Their predictive ability is restricted to those states where their initial conditions are met. In any world where these conditions do not hold the prediction function is undefined.[10]

Defining M: Taking the simplest possible approach, we define M for each OAC as the long-term probability that the OAC's T function correctly predicts the resulting state, assuming the OAC's execution began from a state for which the OAC's prediction function was defined.

[10]We can also imagine OACs whose prediction functions are best defined by disjunctions of separate prediction rules. In such cases, if one of the rules' initial conditions is true, that rule is used to predict the outcome of the action. If no rule matches then the prediction function is undefined.

We note that in classical AI planning systems, the reliability measure for all OACs would be fixed as $M = 1$. Such planners assume a deterministic and totally observable world, thereby removing all uncertainty from their prediction functions. More recent work in AI planning has moved beyond these assumptions (see, e.g., [19, 56]). For instance, there are now a number of planning algorithms that use probabilistic statements about an action's long-term success to build plans with probabilistic bounds on the likelihood that they will achieve their goals. Our definition of M makes our OACs suitable for use by such planners.

Defining E: The execution specifications of these two OAC are straightforward but differ significantly from our previous examples. While each of the previous example OACs indicated a specific control program to execute, our planning OACs define their execution in terms of executing other OACs. For example, the execution specification of **PlanGrasp** is defined in terms of executing **ObjGrasp**:

$$E_{\textbf{PlanGrasp}} = \texttt{execute}(\textbf{ObjGrasp}).$$

This means that invoking `execute`(**PlanGrasp**) calls `execute`(**ObjGrasp**).

Similarly the execution specification for **PlanPush** is defined in terms of our previously defined pushing OAC, **AgnoPush**:

$$E_{\textbf{PlanPush}} = \texttt{execute}(\textbf{AgnoPush}).$$

In other words, `execute`(**PlanPush**) calls `execute`(**AgnoPush**). In Section 8 we will see examples of the execution of these planning-level OACs.

7.4.2. Learning the Prediction Functions of Planning-Level OACs

The problem of learning prediction functions of the form we use in our planning OACs has been the focus of much recent research (see, e.g., [8]). One way this can be done is to use a training set of example actions in the world, and corresponding observations of the world before and after each action. For each example, a reduced world state consisting of a subset of the propositional attributes that make up the entire world model is computed and considered by the learning model. The attribute state is provided as input to the learning model in the form of a vector where each bit corresponds to the value of a single attribute. The learning problem is then treated as a set of binary classification problems, with one classifier for each attribute,

and the model learns the changes to each attribute in the reduced state. The actual learning can be performed, e.g., by using a kernelised voted perceptron classifier [57, 58], which is computationally efficient and can handle noise and partial observability. We refer the reader to [8] for a detailed account of how T and M can be learnt for this kind of OAC.

8. Interacting OACs

In this section, we describe two examples of OACs interacting in a single architecture. In Section 8.1, we illustrate the grounding of objects and object-related grasp affordances. In Section 8.2, we describe how such grounded representations can be used to execute plans.

8.1. Grounding Grasping OACs

In this first example of OAC interaction, we demonstrate the grounding of objects and object-related grasping affordances based on two learning cycles involving the OACs **AgnoGrasp** and **ObjGrasp** (see Figure 10). This process is shown in OAC notation in Algorithm 5 (and was previously described in [35]). The first cycle (Figure 10, top) learns a visual object model of an unknown object by grasping the object using **AgnoGrasp**. Once physical control is achieved, the model can be learnt by integrating the information gained from different views. The second cycle (Figure 10, bottom) learns how to grasp the object. The newly acquired visual model is used to identify and locate the object in the scene. **ObjGrasp** is then used to grasp the object. Through repeated applications of this procedure the performance of **ObjGrasp** is improved.

In this process, object knowledge and grasp knowledge is built up and stored in an internal representation (i.e., the object and grasp memory). Certain characteristics of our OACs play an important role in this process:

- Although the purpose of the first learning cycle is not to learn the OAC **AgnoGrasp** (the aim is to attain physical control over an object), learning is nevertheless taking place by calls to the `updateCP` and `updateM` functions, as a process parallel to those processes steered by, e.g., intentions or automated behaviours. This demonstrates the principle of "ongoing learning" mentioned in Section 3.

- OACs can be combined to produce complex behaviour. The interaction of multiple OACs, as demonstrated in the two learning cycles, can

First learning cycle
while `statusGrasp` $\neq$ `stable` **do**
    open gripper
    `expr` = `execute`(**AgnoGrasp**);
    `updateCP`(`expr`);
    `updateM`(`expr`);
end
Accumulate object representation o_i
if *accumulation successful* **then**
    transfer o_i into object memory $\mathcal{M}^{\mathrm{O}}$
    initialise $\mathbf{ObjGrasp}_{o_i}$ in $\mathcal{M}^{\mathrm{OAC}}$
    Second learning cycle
    while *instance of object* o_i *in scene* **do**
        state.`targetObj` = o_i
        `expr` = `execute`($\mathbf{ObjGrasp}_{o_i}$);
        `updateCP`(`expr`);
        `updateM`(`expr`);
        open gripper
    end
end

Algorithm 5: Grounding of object shape knowledge and object-specific grasp knowledge by cooperative application of the OACs **AgnoGrasp** and **ObjGrasp**.

result in the grounding of symbolic entities usable for planning (see Section 8.2).

8.2. Performing Plans

We now demonstrate how higher-level OACs can be executed by calling lower-level OACs, in the context of performing a plan. To do this, we consider an agent that is given the high-level goal of achieving `inGripper(`o`)` in a world described by the high-level state:

$$\{\texttt{focusOfAttn}(o), \texttt{gripperEmpty}, \neg\texttt{reachable}(o), \\ \texttt{pushable}(o), \texttt{onTable}(o), \texttt{clear}(o)\}.$$

Since `reachable(`o`)` does not initially hold in the world, a high-level planner must build a plan that makes this property true. Using the OACs from Section 7.4, one possible plan is an action sequence that first pushes o into a

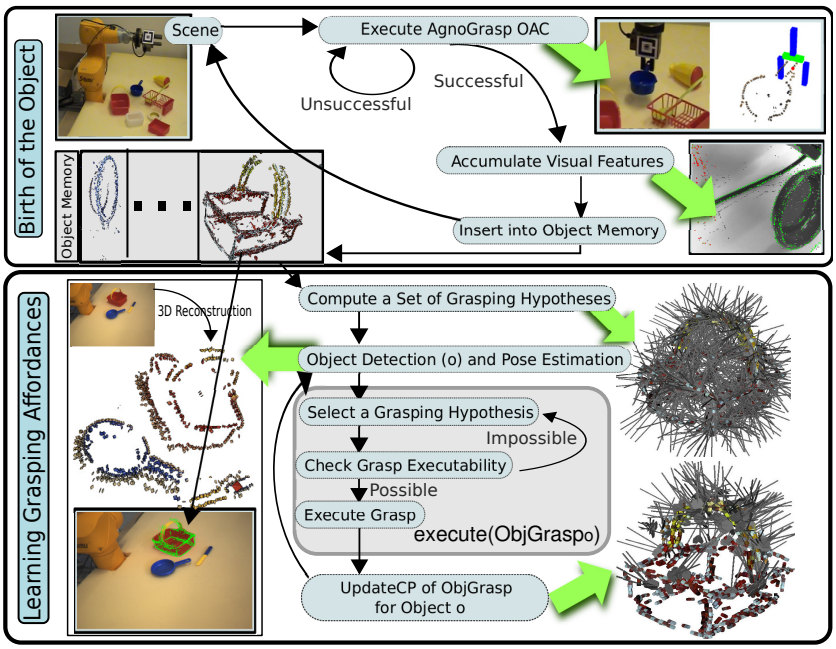


Figure 10: Grounding the OAC **ObjGrasp** in two learning cycles. In the first learning cycle, physical control over a potential object is obtained by the use of **AgnoGrasp**. Once control over the object is achieved and the visual structure changes according to the movement of the robot arm, a 3D object representation is extracted and stored in memory. In the second learning cycle, **ObjGrasp** is established and refined. First, the object representation extracted in the first learning cycle is used to determine the pose of the object in case it is present in the scene. Random samples of these are then tested individually. Successful grasps are turned into a probability density function that represents the grasp affordances associated to the object, in the form of success likelihoods of the grasp parameters.

graspable position, followed by an action that picks up o (see Figure 11).[11] This results in the following plan consisting of the two high-level OACs:

$$\textbf{PlanPush}, \textbf{PlanGrasp}.$$

Recall from Section 6.2 that successful planning using OACs relies on representational congruency and the hierarchical relationship between high-level OACs and lower-level OACs. Further, recall from Section 7.4 that the execution specifications of our high-level OACs are defined in terms of lower-

[11] We refer the reader to [55, 56] for more details on how such planning can be done.

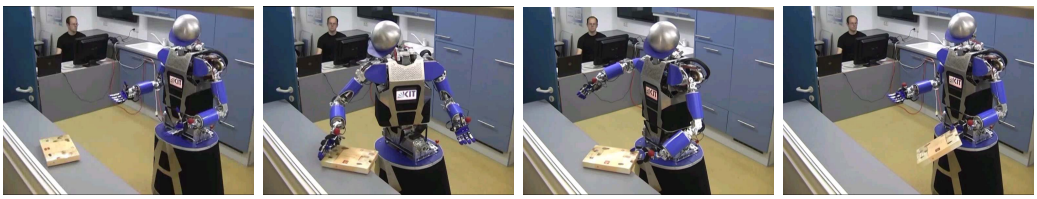


Figure 11: Execution of the plan involving the OACs **PlanPush**, **AgnoPush**, **PlanGrasp**, and **ObjGrasp**. From left to right: (1) the object is not graspable, (2) pushing moves the object into a graspable pose, (3) the object is grasped, and (4) the object can finally be picked up by the agent.

level OACs, so that the execution of a high-level OAC effectively calls a lower-level OAC as a subroutine, i.e.,

$$\begin{aligned} E_{\textbf{PlanPush}} &= \texttt{execute}(\textbf{AgnoPush}), \\ E_{\textbf{PlanGrasp}} &= \texttt{execute}(\textbf{ObjGrasp}). \end{aligned}$$

To understand the execution of the above plan, we must consider the ordering of the respective execution calls—and the experiments returned by those calls—in each of the component OACs in the plan. In this discussion, we assume that the world and the agent act as predicted and planned, without plan or execution failures. For reasons of space, we will also ignore all calls to the associated learning functions. However, even under such simplifying assumptions, the execution of the above plan requires a number of steps. We note that these steps should not be seen as a code fragment but rather as a trace of an executing system:

1. The execution of **PlanPush** is defined in terms of the execution of **AgnoPush**. By our definition of representational congruency, we are guaranteed that information can be translated from **PlanPush**'s high-level representation into **AgnoPush**'s model. For $\texttt{focusOfAttn}(o)$, a process must first be invoked to acquire $\text{bin}(o)$ and extract $\text{loc}(o)$ from the environment. Second, a process must identify τ and a for the desired push operation.
2. As we described in Section 7.1, executing **AgnoPush** invokes a low-level control program that performs the task of actually pushing o, by making use of the agent's end effector.
3. Executing **AgnoPush** returns the experiment:

$$(\{(\text{loc}(o), \text{bin}(o))\},\ \{T(\text{bin}(o), \text{loc}(o), a, \tau)\},\ \{\text{loc}(o)'\})$$

(which can, as all other experiments, either been stored in short term memory or used directly for learning). Representational congruency allows us to use $\text{loc}(o)'$ to determine the truth value of the high-level predicate $\texttt{reachable}(o)$ which is used in the experiment returned by **PlanPush**.[12]

4. Executing **PlanPush** therefore returns the experiment:

$$(\{\neg\texttt{reachable}(o), \texttt{pushable}(o), \texttt{clear}(o), \texttt{gripperEmpty}, \texttt{onTable}(o)\},\\ \{\texttt{reachable}(o)\},\\ \{\texttt{reachable}(o)\})$$

   indicating that $\texttt{reachable}(o)$ is now true in the actual world, and the agent can update its model with this information. This completes the execution of the first action in the plan.
5. The execution of **PlanGrasp** is defined in terms of the execution of $\textbf{ObjGrasp}_o$. As with **PlanPush**, information must be translated from the high-level representation into **AgnoPush**'s model. Since $\texttt{focusOfAttn}(o)$ is true in the world, the translation process, based on representational congruency, ensures that $\texttt{targetObj} = o$.
6. As we described in Section 7.3, executing $\textbf{ObjGrasp}_o$ invokes a low-level control program that performs the task of actually grasping o, by making use of the agent's end effector.
7. Executing $\textbf{ObjGrasp}_o$ returns the experiment:

$$(\{\texttt{statusGripper} = \texttt{empty}, \texttt{targetObj} = o\},\\ \{\texttt{statusGrasp} = \texttt{stable}\},\\ \{\texttt{statusGrasp} = \texttt{stable}\}).$$

   Again, representational congruency ensures $\texttt{statusGrasp} = \texttt{stable}$ can be translated into the attribute space of the higher-level OAC, to determine the truth value of the predicate $\texttt{inGripper}(o)$. This predicate can then be included in the experiment returned by **PlanGrasp**.

[12] We can imagine more complex execution specifications that would monitor the execution of the lower-level pushing OAC and call this OAC repeatedly until $\texttt{reachable}(o)$ is true. In this case, we have assumed that a single push is all that is necessary, and we leave the definition of such complex execution specifications as an area for future work.

8. Executing **PlanGrasp** returns the experiment:

$$(\{\texttt{reachable}(o), \texttt{clear}(o), \texttt{gripperEmpty}, \texttt{onTable}(o)\},$$
$$\{\texttt{inGripper}(o), \neg\texttt{gripperEmpty}, \neg\texttt{onTable}(o)\},$$
$$\{\texttt{inGripper}(o), \neg\texttt{gripperEmpty}, \neg\texttt{onTable}(o)\})$$

indicating that $\texttt{inGripper}(o)$ is now true in the world. As before, the agent can update its high-level model to reflect this fact. This ends the execution of the second action of the plan, and the plan as a whole.

As illustrated in the above example, the successful execution of a plan may typically require OACs to be invoked at multiple levels of abstraction, translating the calls between different models, and monitoring the results to confirm the success of the actions involved. We note that while our definitions support this simple example, more work is needed to extend our formal OAC definitions to more complex control structures for grounding, specifically in the areas of OAC execution, representation congruency, and OAC hierarchies. For instance, if planning is to be effective in real-world domains, *execution monitoring* is essential for detecting divergences of planned states from actual sensed states, and replanning accordingly (see, e.g., [59]). Such processes rely on the structural guarantees that properties like representational congruency provide. We leave the task of generalising such control structures as an area for future work.

9. Conclusion

This paper introduced Object-Action Complexes (OACs) as a framework for modelling actions and their effects in artificial cognitive systems. We provided a formal definition of OACs and a set of concrete examples, showing how OACs operate and interact with other OACs, and also how certain aspects of an OAC can be learnt.

The importance of OACs lies in their ability to combine the properties of multiple action formalisms, from a diverse range of research fields, to provide a dynamic, learnable, refinable, and grounded representation that binds objects, actions, and attributes in a causal model. OACs have the ability to represent and reason about low-level (sensorimotor) processes as well as high-level (symbolic) information and can therefore be used to join the perception-action space of an agent with its planning-reasoning space. In addition, OACs can be combined to produce more complex behaviours,

and sequenced as part of a plan generation process. As a consequence, the OAC concept can be used to bridge the gap between low-level sensorimotor representations, required for robot perception and control, and high-level representations supporting abstract reasoning and planning.

10. Acknowledgements

The research leading to these results received funding from the European Union through the Sixth Framework PACO-PLUS project (IST-FP6-IP-027657) and the Seventh Framework XPERIENCE project (FP7/2007-2013, Grant No. 270273). We thank Frank Guerin for fruitful discussions and his input to Piaget's understanding of sensory-motor schemas.

References

[1] R. A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation 2 (1986) 14–23.

[2] R. A. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, M. M. Williamson, The Cog project: Building a humanoid robot, Lecture Notes in Computer Science 1562 (1999) 52–87.

[3] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, The MIT Press, 1986.

[4] M. Huber, A hybrid architecture for adaptive robot control, Ph.D. thesis, University of Massachusetts Amherst (2000).

[5] A. Stoytchev, Some basic principles of developmental robotics, IEEE Transactions on Autonomous Mental Development 1 (2) (2009) 1–9.

[6] J. Modayil, B. Kuipers, Bootstrap learning for object discovery, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, pp. 742–747.

[7] R. E. Fikes, N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3-4) (1971) 189–208.

[8] K. Mourão, R. Petrick, M. Steedman, Using kernel perceptrons to learn action effects for planning, in: Proceedings of the International Conference on Cognitive Systems, 2008, pp. 45–50.

[9] E. Amir, A. Chang, Learning partially observable deterministic action models, Journal of Artificial Intelligence Research 33 (2008) 349–402.

[10] R. Sutton, Verification, the key to AI, [Online]. Available from: http://www.cs.ualberta.ca/~sutton/IncIdeas/KeytoAI.htm (2001).

[11] S. Harnad, The symbol grounding problem, Physica D (42) (1990) 335–346.

[12] J. Piaget, The Origins of Intelligence in Children, London: Routledge & Kegan Paul, 1936, (French version published in 1936, translation by Margaret Cook published 1952).

[13] F. J. Corbacho, M. A. Arbib, Schema-based learning: Towards a theory of organization for biologically-inspired autonomous agents, in: Proceedings of the First International Conference on Autonomous Agents, 1997, pp. 520–521.

[14] A. Newell, H. Simon, GPS, a program that simulates human thought, in: E. A. Feigenbaum, J. Feldman (Eds.), Computers and Thought, McGraw-Hill, NY, 1963, pp. 279–293.

[15] C. Green, Application of theorem proving to problem solving, in: Proceedings of the First International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1969, pp. 741–747.

[16] E. D. Sacerdoti, The nonlinear nature of plans, in: Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1975, pp. 206–214.

[17] A. Samuel, Some studies in machine learning using the game of checkers, IBM Journal of Research and Development 3 (3) (1959) 210–229.

[18] N. J. Nilsson, Learning Machines, McGraw-Hill, 1965.

[19] L. P. Kaelbling, Learning functions in k-DNF from reinforcement, in: Proceedings of the Seventh International Workshop on Machine Learning, Morgan Kaufmann, 1990, pp. 162–169.

[20] T. Mitchel, Machine Learning, WCB McGraw Hill, 1997.

[21] H. Pasula, L. Zettlemoyer, L. P. Kaelbling, Learning symbolic models of stochastic domains, Journal of Artificial Intelligence 29 (2007) 309–352.

[22] F. Wörgötter, A. Agostini, N. Krüger, N. Shyloa, B. Porr, Cognitive agents — a procedural perspective relying on the predictability of object-action-complexes (OACs), Robotics and Autonomous Systems 57 (4) (2009) 420–432.

[23] D. Vernon, G. G. Metta, G. Sandini, A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents, IEEE Transactions on Evolutionary Computation 11 (2007) 151–180.

[24] L. P. Kaelbling, M. L. Littman, A. R. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Intell. 101 (1998) 99–134. doi:10.1016/S0004-3702(98)00023-X.
URL http://portal.acm.org/citation.cfm?id=1643275.1643301

[25] J. Pearl, Probabilistic Reaasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.

[26] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likehood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society (Series B) 39 (1) (1977) 1–38.

[27] D. Spiegelhalter, P. Dawid, S. Lauritzen, R. Cowell, Bayesian analysis in expert systems, Statistical Science 8 (1993) 219–283.

[28] S. Kok, P. Domingos, Learning the structure of Markov logic networks, in: Proceedings of the Twenty-Second International Conference on Machine Learning, ACM Press, 2005, pp. 441–448.

[29] J. J. Gibson, The Perception of the Visual World, Houghton Mifflin, Boston, 1950.

[30] E. Sahin, M. Çakmak, M. R. Doğar, E. Uğur, G. Ücoluk, To afford or not to afford: A new formalization of affordances toward affordance-based robot control, Adaptive Behavior 15 (4) (2007) 447–472.

[31] R. Brooks, Intelligence without reason, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1991, pp. 569–595.

[32] R. Brooks, Elephants don't play chess, Robotics and Autonomous Systems 6 (1&2) (1990) 3–15.

[33] R. Pfeifer, M. Lungarella, F. Iida, Self-organization, embodiment, and biologically inspired robotics, Science 318 (2007) 1088–1093.

[34] D. Kraft, N. Pugeault, E. Başeski, M. Popović, D. Kragic, S. Kalkan, F. Wörgötter, N. Krüger, Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes, Special Issue on "Cognitive Humanoid Robots" of the International Journal of Humanoid Robotics 5 (2009) 247–265.

[35] D. Kraft, R. Detry, N. Pugeault, E. Başeski, , F. Guerin, J. Piater, N. Krüger, Development of object and grasping knowledge by robot exploration, IEEE Transactions on Autonomous Mental Development 2 (4) (2010) 368–383.

[36] A. Stoytchev, Behavior-Grounded Representation of Tool Affordances, in: IEEE International Conference on Robotics and Automation, 2005, pp. 3060–3065.

[37] R. Jacobs, M. Jordan, A. Barto, Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks, Cognitive Science 15 (2) (1991) 219–250.

[38] C. M. Vigorito, A. G. Barto, Intrinsically motivated hierarchical skill learning in structured environments, IEEE Transactions on Autonomous Mental Development (TAMD) 2 (2) (2010) 132–143.

[39] K. Narendra, J. Balakrishnan, Adaptive control using multiple models, IEEE Transaction on Automatic Control 42 (2) (1997) 171–187.

[40] M. Haruno, D. Wolpert, M. Kawato, MOSAIC model for sensorimotor learning and control, Neural Computation 13 (2001) 2201–2220.

[41] C. Miall, Modular motor learning, Trends in Cognitive Sciences 6 (1) (2002) 1–3.

[42] J. Gibson, The Ecological Approach to Visual Perception, Boston, MA: Houghton Mifflin, 1979.

[43] R. Detry, D. Kraft, A. G. Buch, N. Krüger, J. Piater, Refining grasp affordance models by experience, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2010, pp. 2287–2293.

[44] T. A. Henzinger, The theory of hybrid automata, in: M. Inan, R. Kurshan (Eds.), Verification of Digital and Hybrid Systems, Vol. 170 of NATO ASI Series F: Computer and Systems Sciences, Springer, 2000, pp. 265–292.

[45] A. D. Baddeley, Essentials of Human Memory, Psychology Press, Taylor and Francis, 1999.

[46] D. Willshaw, P. Buneman, C. Longuet-Higgins, Non-Holographic Associative Memory, Nature 222 (1969) 960–962.

[47] D. Willshaw, Holography, Association and Induction, in: G. Hinton, J. Anderson (Eds.), Parallel Models of Associative Memory, Erlbaum, Hillsdale, NJ, 1981, pp. 83–104.

[48] F. T. Sommer, G. Palm, Bidirectional Retrieval from Associative Memory, Advances in Neural Information Processing Systems 10 (1998) 675–681.

[49] T. Plate, Holographic Reduced Representations: Convolution Algebra for Compositional Distributed Representations, in: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1991, pp. 30–35.

[50] D. Omrčen, A. Ude, A. Kos, Learning primitive actions through object exploration, in: Proceedings of the International Conference on Humanoid Robots, Daejeon, Korea, 2008, pp. 306–311.

[51] M. Popović, D. Kraft, L. Bodenhagen, E. Başeski, N. Pugeault, D. Kragic, T. Asfour, N. Krüger, A strategy for grasping unknown objects based on co-planarity and colour information, Robotics and Autonomous Systems 58 (5) (2010) 551–565.

[52] L. Bodenhagen, D. Kraft, M. Popović, E. Başeski, P. E. Hotz, N. Krüger, Learning to grasp unknown objects based on 3D edge information, in: Proceedings of the IEEE International Conference on Computational Intelligence in Robotics and Automation, 2009, pp. 421–428.

[53] N. Pugeault, F. Wörgötter, N. Krüger, Visual primitives: Local, condensed, semantically rich visual descriptors and their applications in robotics, Special Issue on "Cognitive Humanoid Vision" of the International Journal of Humanoid Robotics 7 (3) (2011) 379–405.

[54] R. Detry, N. Pugeault, J. Piater, A probabilistic framework for 3D visual object representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (10) (2009) 1790–1803.

[55] R. Petrick, D. Kraft, K. Mourão, C. Geib, N. Pugeault, N. Krüger, M. Steedman, Representation and integration: Combining robot control, high-level planning, and action learning, in: Proceedings of the Sixth International Cognitive Robotics Workshop, Patras, Greece, 2008, pp. 32–41.

[56] R. Petrick, F. Bacchus, A knowledge-based approach to planning with incomplete information and sensing, in: Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling, 2002, pp. 212–221.

[57] Y. Freund, R. Schapire, Large margin classification using the perceptron algorithm, Machine Learning 37 (1999) 277–296.

[58] R. Khardon, G. M. Wachman, Noise tolerant variants of the perceptron algorithm, Journal of Machine Learning Research 8 (2007) 227–248.

[59] R. Petrick, D. Kraft, N. Krüger, M. Steedman, Combining cognitive vision, knowledge-level planning with sensing, and execution monitoring for effective robot control, in: Proceedings of the Fourth Workshop on Planning and Plan Execution for Real-World Systems, Thessaloniki, Greece, 2009, pp. 58–65.

Norbert Krüger is a Professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. He holds an M.Sc. from the Ruhr-Universität Bochum, Germany and his Ph.D. from the University of Bielefeld. Norbert Krüger leads the Cognitive Vision Lab which focuses on computer vision and cognitive systems, in particular the learning of object representations in the context of grasping. He has also been working in the areas of computational neuroscience and machine learning.

Christopher Geib is a Research Fellow at the University of Edinburgh School of Informatics. He holds an M.S. and Ph.D. from the University of Pennsylvania. His research focuses broadly on decision making and reasoning about actions under conditions of uncertainty, including planning, scheduling, constraint-based reasoning, human-computer interaction, human-robot interaction, and probabilistic reasoning. His recent research has focused on probabilistic intent recognition through weighted model counting and planning based on grammatical formalisms.

Justus Piater is a professor of computer science at the University of Innsbruck, Austria. He earned his Ph.D. degree at the University of Massachusetts Amherst, USA, where he held a Fulbright graduate student fellowship. After a European Marie-Curie Individual Fellowship at INRIA Rhône-Alpes, France, he was a professor at the University of Liège, Belgium, and a Visiting Research Scientist at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany. His research in computer vision and machine learning is motivated by intelligent and interactive systems, where he focuses on visual learning, closed-loop interaction of sensorimotor systems, and video analysis.

Ronald Petrick is a Research Fellow in the School of Informatics at the University of Edinburgh. He received an M.Math. degree in computer science from the University of Waterloo and a Ph.D. in computer science from the University of Toronto. His research interests include planning with incomplete information and sensing, cognitive robotics, knowledge representation and reasoning, generalised planning, and natural language dialogue. He is currently the Scientific Coordinator of the EU JAMES project.

Mark Steedman is a Professor of Cognitive Science in Informatics at the University of Edinburgh, working in computational linguistics, artificial intelligence, the communicative use of prosody, tense and aspect, and wide-coverage parsing using Combinatory Categorial Grammar (CCG). Prof. Steedman is a Fellow of the Association for the Advancement of Artificial Intelligence (AAAI), the Royal Society of Edinburgh (FRSE), and the British Academy (FBA). He is a member of the Academy of Europe and a former President of the Association for Computational Linguistics (ACL).

Florentin Wörgötter has studied Biology and Mathematics in Düsseldorf. He received his Ph.D. in 1988 in Essen working experimentally on the visual cortex before he turned to computational issues at the Caltech, USA (1988-1990). After 1990 he was researcher at the University of Bochum concerned with experimental and computational neuroscience of the visual system. Between 2000 and 2005 he had been Professor for Computational Neuroscience at the Psychology Department of the University of Stirling, Scotland where his interests strongly turned towards "Learning in Neurons". Since July 2005 he leads the Department for Computational Neuroscience of the Bernstein Center at the University of Göttingen. His main research interest is information processing in closed-loop perception-action systems, which includes aspects of sensory processing, motor control and learning/plasticity. These approaches are tested in walking as well as driving robotic implementations. His group has developed the RunBot a fast and adaptive biped walking robot.

Aleš Ude studied applied mathematics at the University of Ljubljana, Slovenia, and received his doctoral degree from the Faculty of Informatics, University of Karlsruhe, Germany. He was awarded the STA fellowship for postdoctoral studies in ERATO Kawato Dynamic Brain Project, Japan. He has been a visiting researcher at ATR Computational Neuroscience Laboratories, Kyoto, Japan, for a number of years and is still associated with this group. Currently he is a senior researcher at the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. His research focuses on imitation and action learning, perception of human activity, humanoid robot vision, and humanoid cognition.

Tamim Asfour received his diploma degree in electrical engineering and his Ph.D. degree in computer science from the University of Karlsruhe, Germany in 1994 and 2003, respectively. He is leader of the humanoid robotics research group at the institute for Anthropomatics at the Karlsruhe Institute of Technology (KIT). His research interests include humanoid robotics, grasping and manipulation, imitation learning, system integration and mechatronics.

Dirk Kraft obtained a diploma degree in computer science from the University of Karlsruhe (TH), Germany in 2006 and a Ph.D. degree from the University of Southern Denmark in 2009. He is currently employed as an assistant professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. His research interests lie within cognitive systems, robotics and computer vision.

Damir Omrčen received his PhD in robotics from the University of Ljubljana, Slovenia, in 2005. He is employed as a research assistant at the Department of Automation, Biocybernetics and Robotics at the "Jozef Stefan" Institute in Ljubljana. His fields of interest include vision and robot control where he combines classical model-based approaches and more advanced approaches based on exploration and learning.

Alejandro Agostini received the B.S. degrees in Bioengineering with honours from the National University of Entre Ríos, Argentina, and in Electronic Engineering from the National University of Catalonia (UPC), Spain. He is currently a senior Ph.D. student in Artificial Intelligence at the UPC, and is working at the Institut de Robòtica Industrial (IRI) under a work contract drawn up by the Spanish Research Council (CSIC). His research interests include machine learning, robotics, decision making, and cognitive systems. He performed several research stays at the Bernstein Centre for Computational Neuroscience, Göttingen, Germany, and at the University of Karlsruhe, Germany.

Rüdiger Dillmann is Professor at the Computer Science Faculty, Karlsruhe Institute of Technology (KIT), Germany. He is director of the Research Center for Information Science (FZI), Karlsruhe. He is scientific leader of German collaborative research centre Humanoid Robots (SFB 588). His research interests include humanoid robotics, technical cognitive systems, machine learning, computer vision, robot programming by demonstration, and medical applications of informatics.

Published at
Convention AISB 2011('Artificial Intelligence and Simulation of Behaviour'), York, 4-7 April 2011

Grasp Learning by Means of Developing Sensorimotor Schemas and Generic World Knowledge

Norbert Kruger[1] and **Mila Popovic**[1] and **Leon Bodenhagen**[1] and **Dirk Kraft**[1] and **Frank Guerin**[2]

Abstract. We present a cognitive system in which grasping competences are coded by means of a formalization of sensory motor schemas in terms of so called 'object action complexes' (OACs). OACs define the knowledge of the system via the effects and precondition of certain behavioural patterns, and also code the uncertainty associated with their execution. OACs are grounded through the observation and evaluation of individual executions generating 'experiments', and dynamically adapt through using these experiments for learning. Moreover, in parallel with the development and refinement of OACs, generic world knowledge is permanently generated by the system which affects the OACs on a meta level and provides a means for the generation of new competences and better generalization. We present an example of a developing system executing OACs which code the grasping of known and unknown objects, and thereby illustrate (i) the refinement of OACs and (ii) building up generic world knowledge. We see this as particularly important since these interaction processes, although fundamental for human development, are usually difficult to observe by means of techniques in neurophysiology and developmental psychology.

1 Introduction

Cognitive development seems to proceed at a number of different levels of abstraction: for example there are low level developments such as perception-action control loops for basic sensorimotor skills involved in reaching, grasping, object manipulation, and walking; in parallel with this there are higher developments in the knowledge of objects, physical causality, and spatial relationships (these are more abstract than the lower level, and allow for application in a range of scenarios). There is a strong connection between these parallel tracks; higher-level knowledge seems to be abstracted from lower level context specific sensorimotor routines (see e.g. [31]), and seems to arrive after the acquisition of skills at the lower level. In the other direction, the higher level knowledge, once attained, can be used to improve the appropriate application and adjustment of lower level skills (see e.g. Piaget on the "support" [24, 25]). A major challenge is to explain (mechanistically) how this parallel development works. Such an explanation seems to be necessary in order to understand how such advanced abilities as tool-use develop ontogenetically in humans; contemporary opinion in psychology holds that advanced tool-use has its origins in infants' early exploratory interactions with objects and surfaces, and that the development from these precursors to advanced manipulations is gradual and continuous [19].

[1] Syddansk Universitet, Maersk Mc-Kinney Institute, DK-5230 Odense, Denmark.
[2] University of Aberdeen, Department of Computing Science, Aberdeen AB24 3UE, Scotland.

In this paper we tackle one fragment of this development: that is the fragment related to grasping; at lower levels of abstraction we can learn specific sensorimotor routines for grasping specific objects, but at a higher level (and in parallel) we can learn more generic object knowledge which can improve the grasping of known objects, and also help us to grasp novel objects. More specifically we capture the sensorimotor skill of grasping within the framework of Object Action Complexes (OACs) (a formal framework introduced in [18, 38]). The formalism of OACs is a skeleton — which integrates existing concepts in the field of artificial intelligence as well as (behavioural and) cognitive robotics (see Sec. 2) — that can be used to formalize adaptive and predictive behaviours on different levels of the processing hierarchy. OAC executions generate empirical data in terms of so called 'experiments', and these lead to different kinds of learning which are clearly distinguished. This learning ensures grounding and leads to an ongoing improvement of the overall system through adaptation and learning. By that OACs should be able to reach from low level reactive actions to conscious planning through the experience of actions applied to objects in the world (for details, see [18]). In this paper, we will use this OAC concept to outline a framework for the development of sensorimotor skills associated with grasping as well as the parallel development of generic world knowledge.

As one innate grasping mechanism we make use of simple manually defined feature–action associations (see [27]) triggered by the early cognitive vision system [29]. These association are motivated by innate 'grasping reflexes' in infants although they differ in detail due to different embodiments (see [17] for a discussion of similarities and differences to infant's grasping). This initial 'grasping reflex' is coded as an object action complex OAC^{gen}. It becomes refined during its application in the exploration process through learning. In a process (described in detail in [17]) triggered by OAC^{gen}, world knowledge in terms of object shape knowledge is extracted. Once this shape knowledge is available to the system, object specific grasp knowledge is learned and coded in terms of a second OAC OAC_o^{grasp}. While OAC^{gen} codes generic feature grasp associations, OAC_o^{grasp} associates grasp knowledge with a specific learned object o based on the concept of grasp densities as outlined in [10]. OAC_o^{grasp} and OAC^{gen} code two different strategies associated with different branches of grasp research. While OAC^{gen} codes generic grasp affordances (see, e.g.,[30, 8, 5, 27] for work related to generic grasping), OAC_o^{grasp} addresses the grasping of specific objects (see, e.g., [4, 20, 15, 10]).

In this paper, we outline how these two kinds of OACs develop in parallel in a cognitive system, generate generic world knowledge and in particular support each other by making use of this developing generic world knowledge. This builds on existing work where we have shown that

- Innately defined feature action associations can already lead to rather high performance grasping [27].
- Both OACs are refined over time through learning processes associated with the OACs individually [27, 10].
- OAC^{gen} can be used to initiate the developmental process of OAC_o^{grasp} [17].

Building on this background, the current paper shows that convergence speed is a fundamental problem associated with OAC_o^{grasp} which is basically 'learning by heart' with only little generalisation by means of local interpolation (See Sec. 5.2). Furthermore, on the strength of our first experimental indications described here, we make the following speculative predictions

- OAC_o^{grasp} delivers the statistical material for the branching of OAC^{gen} into new OACs expressing new grasping affordances. This is done by finding indicative feature relations–grasp association in co-occurrence tables coding visual feature relation and the grasping success associated with grasps related to them (see Sec. 6.2).
- OAC_o^{grasp} and OAC^{gen} deliver the statistical material to fundamentally change the learning algorithm associated with OAC_o^{grasp} and by that lead to a faster convergence of OAC_o^{grasp}. This is done by using the co-occurrence statistics to refine kernels in the KDE approach [32] applied in the grasp density concept (See Sec. 6.1).
- Finally, at the end of the developmental process, OAC^{gen} (coding grasping without object knowledge) eventually becomes powerful enough to generate grasp densities close to the ones which are initially tediously learned by OAC_o^{grasp}, hence that there is no fundamental difference between grasping known and unknown objects anymore.

This paper will partly refer to already published work [27, 10, 17] while putting it in a developmental context, partly refer novel and ongoing work with new (and to a certain degree intermediate) results and partly make speculative predictions based on available data. The aim is to put existing and ongoing work in a wider context addressing the fundamental problem of learning of sensory motor schemas for tool use in an embodied robot system.

2 Sensorimotor schemas and object action complexes

The sensorimotor schema[3] as defined by Piaget and others [26, 9] is a dynamic entity that gathers together the perceptions and associated actions involved in the performance of a habitual behaviour. The schema represents knowledge generalized from all the experiences of that behaviour. It also includes knowledge about the context in which the behaviour was performed as well as expectations about the effects. During cognitive development these schemas are refined and combined. Object Action Complexes (OACs) are a formalisation of such schemas to be used in artificial cognitive systems (see [18]).

An OAC's definition is split into three parts, (1) a *symbolic description* consisting of a prediction function defined over an attribute space, together with a measure of the reliability of the OAC, and (2) an *execution specification* that defines how the OAC is executed by the embodied system and generates experience in terms of 'experiments' and (3) a specification of how the learning associated with

[3] also called "sensorimotor process" [33], "skill" [13], or "perception-action routine" [19].

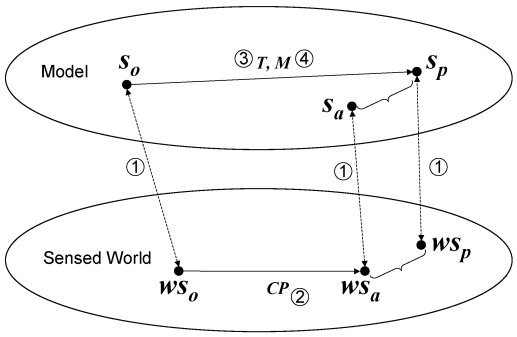


Figure 1: Graphical representation of an OAC and the OAC learning problems. This shows how the actual state ws_a (corresponding to s_a in the model) resulting from the execution of the control program *CP* diverges from the state s_p predicted by the OAC's prediction function T. This divergence drives the learning (i.e. refinement) of the OAC. For further explanation see text. Figure courtesy of Christopher Geib.

the OAC is realized based on the 'experiments' generated by the executed OACs. More formally (see also Fig. 1):

Definition 2.1 *An **Object-Action Complex (OAC)** is a triple*

$$(\mathtt{id}, T, M) \tag{1}$$

where:

- $\mathtt{id}$ *is a unique identifier for an execution specification,*
- $T : \mathcal{S} \rightarrow \mathcal{S}$ *is a prediction function defined on an attribute space $\mathcal{S}$ encoding the system's beliefs as to how the world (and the robot) will change if the control is executed , and*
- M *is a statistical measure representing the success of the OAC within a window over the past.*

An execution function `execute` (see below) can map an OAC `id` to an 'experiment' which is defined the following way:

Definition 2.2 *Given an attribute space $\mathcal{S}$ and an OAC with identifier* $\mathtt{id}$ *defined on $\mathcal{S}$, an **experiment** is a triple*

$$(s_o, s_p, s_a) \tag{2}$$

where:

- $s_o \in \mathcal{S}$*, captures the state of world before execution*
- $s_p \in \mathcal{S}$ *such that OAC* $\mathtt{id}$ *predicts that state s_p will result from its execution in s_o, i.e., $s_p = T_{\mathtt{id}}(s_o)$, and*
- $s_a \in \mathcal{S}$ *such that s_a is observed as a result of actually executing OAC* $\mathtt{id}$ *in state s_o.*

Thus, an experiment is an *empirical event* by which OACs will be grounded in sensory experience.

As an empirical grounded event, such experiments can be used to update OACs in cycles of execution and learning based on evaluations of their success (see below). Note that sometimes an experiment is actually not used directly for learning but stored in some short term memory (see, e.g., [3]) until resources for learning are available (e.g., during 'sleeping phases').

The execution, i.e., the actual action associated to the OAC is defied as following:

Definition 2.3 `execute` *is a function that maps an OAC* `id` *to an experiment, i.e.,*

$$\texttt{execute} : id \rightarrow \texttt{expr} = (s_o, s_p, s_a). \quad (3)$$

The `execute` function is an operation that performs the OACs execution specification in the current world state, returning an experiment `expr`.

The definition of OACs as capturing both symbolic and control knowledge about actions highlights a number of learning problems that must be addressed for OACs to be effective. We note that while each of these learning problems can be addressed by recognizing the differences between predicted states and those states actually achieved, they may still require different learning algorithms (e.g., Bayesian, neural network-like, parametric, non-parametric, etc.). It is up to the OAC designer to choose an appropriate learning mechanism.

As such, the following characterizations are intended to specify those aspects of the OAC that are modified through learning, not the method of learning. We consider four main learning problems, each of which is labelled by its respective number in Fig. 1. These are (1) learning control programs, (2) learning the prediction function, (3) learning the mapping from states of the real world to states of the model and (4) learning the prediction function's long term statistics. In our context, it is only the learning problems (1) and (4) which are of relevance, these are referred to by `updateCP` and `updateM` respectively. All learning functions take an experiment as an argument, e.g., `updateCP(expr)`.

3 Overview: Developmental process in a cognitive architecture

As mentioned in Sec. 1, cognitive development seems to proceed at a number of different levels of abstraction. Fig. 2 shows two such parallel tracks of development. On the bottom is the sensorimotor track which shows the development of lower level sensorimotor schemas (SMSs), which are observable in infant behaviour. Each node in the lower track corresponds to an SMS. A directed edge travels from each ancestor node to its descendents; for example the SMS for pulling a string descends from a basic grasping SMS. Some SMSs have more than one ancestor; for example an infant means-end behaviour will have as ancestors one SMS for the means and one for the goal. The top of Fig. 2 is the abstract track which shows the parallel development of the underlying world knowledge. Nodes in the upper track correspond to (for example) fragments of object knowledge which are common to a number of SMSs, and fragments of spatial relationships; these are gradually linked up as development progresses (to the right), to eventually form a more comprehensive knowledge of objects and space. We must stress here that the early fragments of object and spatial knowledge are likely to be very context specific, and strongly associated with the sensorimotor schemas they have been abstracted from. It is only after a long devlomental process moving to the right in Fig. 2) that these fragments become more objective, and this developmental process must involve some sort of "representational redescription" [7]. The evidence from the psychology literature suggests that it is doubtful that very much objective knowledge is achieved during infancy, but rather that a high degree of context specificity persists [34, 36, 1, see for example].

For the lower track we see a developmental process in which a small set of innately defined SMSs lead to a large variety of SMSs through branching and specialisation. During this developmental process, the effects of the SMS become increasingly predictable and

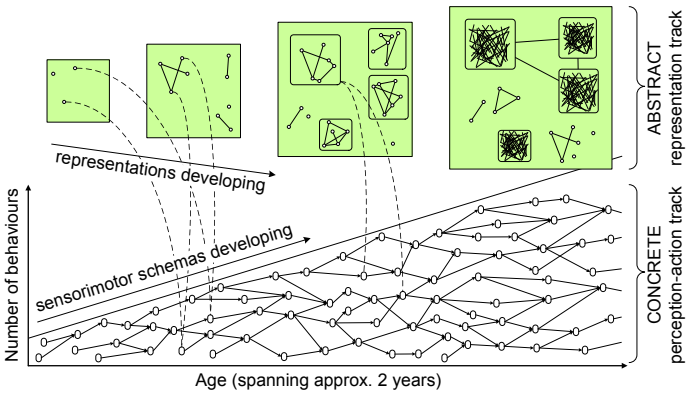


Figure 2: Conceptual diagram, overviewing infant developments on both a low level sensorimotor track and a higher level representational track; for explanation see text (Sec. 3).

can then be used more and more purposefully by the cognitive agent for the planning of behaviour. In parallel to (and triggered by) the development of individual SMSs more generic world knowledge is built up; as illustrated in the upper track of Fig. 2. This is done through the abstraction of empirical data gained during the execution of the SMSs on objects and associated actions. The central topic of this paper is the parallel development and interaction of observable sensorimotor schemas and the increasing abstract world knowledge which is based on the experiments generated by the OACs.

In our case an "innate" SMS is a generic grasping OAC (which would correspond to a single node to the lower left of Fig. 2). This OAC then branches as different objects are encountered, spinning off a new specific grasping OAC for each new object (in this paper three example objects are tackled). On the upper track we have representations of objects which are acquired in "object memory" (see Fig. 7), and also generic knowledge about the relations between low-level visual features and the existence of grasping affordances.

Figure 2 also illustrates (with dashed curves) links between the abstract and sensorimotor tracks; these links are bidirectional. To avoid clutter only six links are shown, but in reality all representational fragments will be linked to sensorimotor schemas. In one direction representations are linked to the schemas they have been generalised from (and hence can immediately link to actions which can manipulate the represented object or spatial relation). In our grasping system this means that the object representations, and general feature-grasping relationships have been abstracted from lower level interactions. In the other direction, more advanced schemas make use of the newly formed representations, for example in their description of the context in which a behaviour may be performed, or its effects, or the control policy followed during execution of the schema. In our grasping system this means that (i) the grasping of the three specific objects which the system has practised on will be able to make use of this more abstract knowledge once it is available, so the grasp success rate will be much higher once sufficient statistics are gathered on the general relationships between visual features and grasp success; (ii) the grasping of novel objects, using the generic grasp OAC will also leverage this generic knowledge leading it to also have a high degree of success.

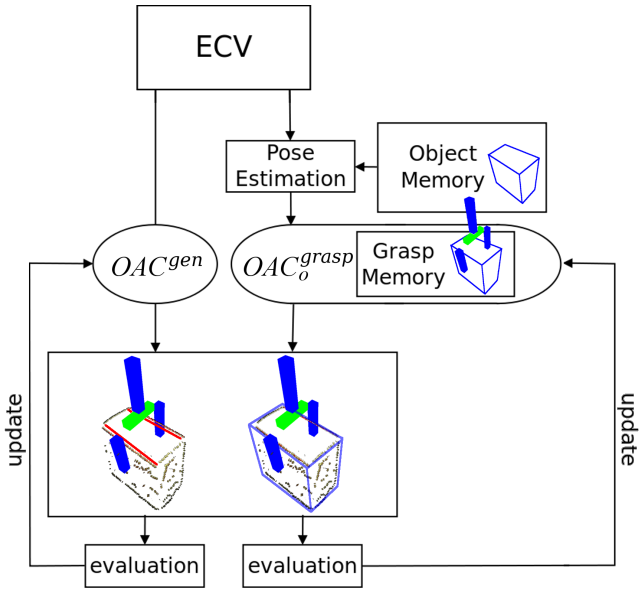


Figure 3: System architecture, see section 4.1

4 Formalising grasping with and without object knowledge: OAC_o^{grasp} and OAC^{gen}

Grasping novel objects is one important example of sensorimotor schemas (SMS) (see, e.g., [26, 21]). An important property of an SMS is that it becomes grounded, refined and sometimes significantly modified during the developmental process. In this section we present the two OACs which formalize two sensorimotor schema associated with grasping, with and without object knowledge. Before we describe these two OACs in more detail in section 4.2 and 4.3, we give some basic information on the robot vision system in which the developmental process is taking place in section 4.1.

4.1 System in which development is taking place

In the system we envisage, the grasping process is organized as sketched in Fig. 3. The two OACs, OAC_o^{grasp} and OAC^{gen}, follow different paths. OAC^{gen} is based on combination of visual features computed by the early cognitive vision (ECV) system (described below). The output of the ECV system is used directly for producing grasping hypotheses (see also Fig. 11). In case of OAC_o^{grasp} the acquired image representation is compared against a database of stored object models, and once the pose estimation is done it is possible through OAC_o^{grasp} to access abstracted grasp knowledge for the known objects in the scene (see section 4.3). Suggested grasping hypotheses can be tested (both in simulation or with the real setup) and the results are used to continuously improve OAC_o^{grasp} and OAC^{gen}.

The visual representation extracted by the early cognitive vision (ECV) system [29] provides rich visual representations for edge structures, surfaces and junctions. Sparse 2D and 3D features, so-called *multi-modal primitives*, are created along image contours and textured areas. These 2D features represent a small image patch in terms of position, orientation and also appearance information (e.g., colour and phase). Primitives describing edge patches are called line segments, primitives describing textured surfaces are called texlets and primitives describing corners (intersections of edges) are called junctions. 2D primitives are matched across two stereo views, and pairs of corresponding 2D features permit the reconstruction of a 3D

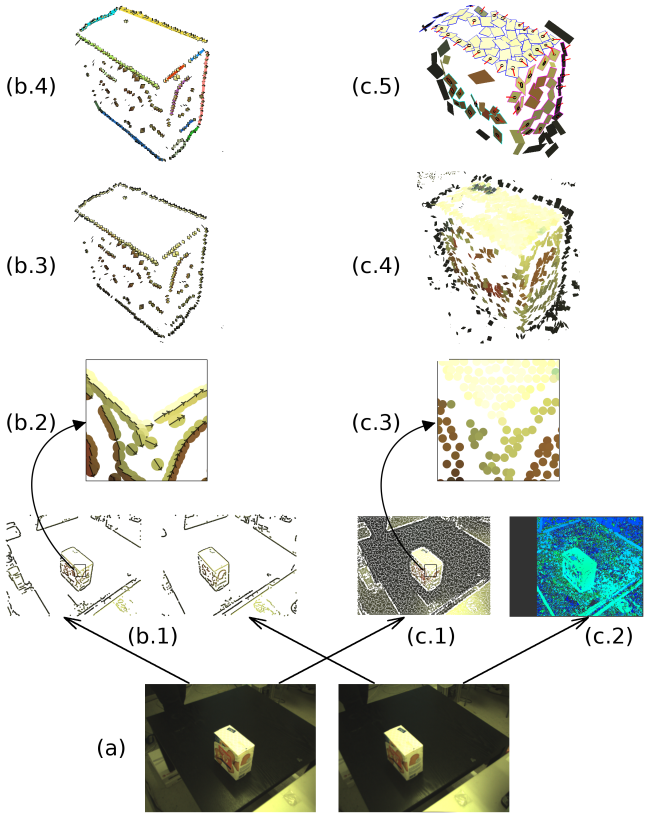


Figure 4: (a) an example stereo image pair. (b.1) 2D line segments for the left and the right image. (b.2) a detail from b.1. (b.3) 3D line segments. (b.4) 3D contours. (c.1) 2D texlets for the left image. (c.2) disparity image. (c.3) a detail from c.1. (c.4) 3D texlets. (c.5) 3D surflings.

equivalent. The 2D and 3D primitives are organized into perceptual groups in 2D and 3D (called contours for line segments, or surflings for the texlets). The procedure to create visual representations (line segments and texlets) is illustrated in Fig. 4 on an example stereo image pair.

The sparse and symbolic nature of the multi-modal primitives allows for the coding of relevant perceptual structures that express relevant spatial relations in 2D and 3D [2]. The relations between contours (and also surflings) allow us to define grasping hypotheses (see section 4.2 and Fig. 11).

4.2 OAC^{gen}: Grasp affordances as feature relation - action associations

OAC^{gen} is used to gain physical control over unknown objects, a grasp computation mechanism based on previous work [27] is used. Pairs of 3D contours that share a common plane and have similar colours suggest a possible grasp (see Fig. 5b). The grasp location is defined by the position of one of the contours. Grasp orientation is calculated from the common plane defined by the two contours and the contour's orientation at the grasp location.

During execution, grasping hypotheses from co-planar contour pairs are computed. The initial attribute space is given by

$$s_o = (|\Omega|,\ (C_1, C_2), \texttt{statusGrasp}),$$

where $|\Omega|$ is the number of elements in the set Ω, and (C_1, C_2) is the concrete pair of extracted contours that was picked earlier. Recall that

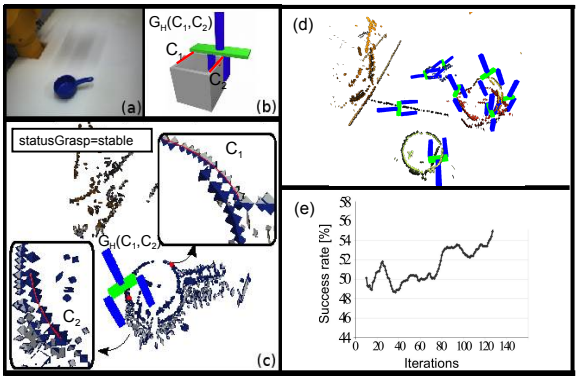


Figure 5: (a) The image of the scene captured by the left camera. (b) A possible grasping action type defined by using the two coplanar contours C_1 and C_2 shown in red. (c) A successful grasping hypothesis. The 3D contours from which the grasp was calculated are shown. Note that the information displayed is the core of an "`expr`". (d) A selected set of grasping hypotheses generated for a similar scene. (e) Change of performance as a result of the learning process.

$OAC^{gen} = (\texttt{id}, T, M)$. The prediction function T in our context is trivial, since a stable grasp ($\texttt{statusGrasp} = \mathit{stable}$) is predicted. M measures the percentage of successful grasps in a certain time window (see Fig. 5e).

The computed grasping hypothesis is then performed and the grasp status $s_a = \texttt{statusGrasp}_{t+1}$ is sensed after picking up the object, resulting in an experiment (see Fig. 5c for the main components of an experiment):

$$\texttt{expr} = \{s_o, \texttt{statusGrasp}_{t+1} = \texttt{stable}, \texttt{statusGrasp}_{t+1}\}.$$

Each experiment can either be used directly for on-line learning, as in the following learning cycle:

while *true* **do**
  choose pair of contours C_1, C_2
  `expr=execute(`gen`);`
  `updateCP(expr);`
  `updateM(expr);`
  drop object
end

or stored in an episodic memory for off-line learning at a later stage by calling the function `updateCP` (see [27] for details). There we have shown that, based on these labelled experiences, we can learn an improved feature-based grasp generation mechanism. `updateCP` uses an artificial neural net to determine which feature relations predict successful grasps. Fig. 5e shows how the success rate increases when on-line learning is performed on the evaluated grasps. The learning is limited by the amount of grasp data available and by the noise that is present in the data. However, as the objects are unknown by the system, the performance is not expected to increase to nearly 100 % even if unlimited training data would be available.

4.3 OAC_o^{grasp}: Object specific grasping

OAC_o^{grasp} expresses affordance relative object-gripper configurations that yield stable grasps. The grasps we consider are parameterised by a 6D gripper pose composed of a 3D position and a 3D

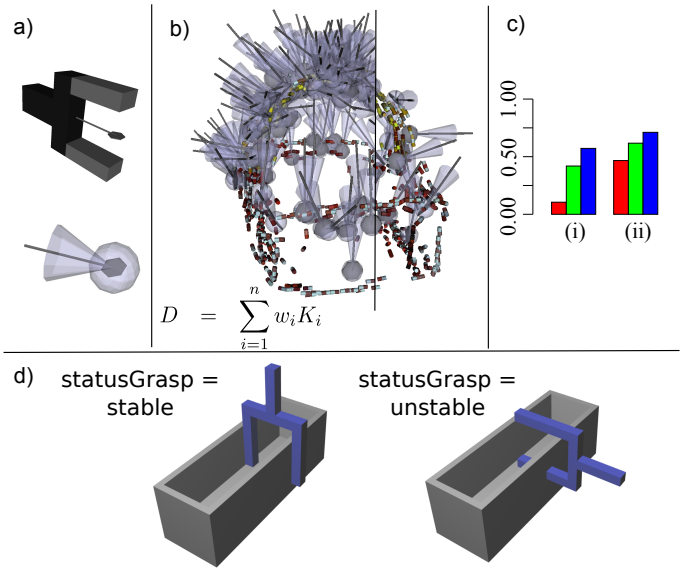


Figure 6: Grasping affordances are represented using kernel-based grasp densities. a) Iso-probable surface of a 'grasp kernel', and relation between a two-finger grasp and a kernel representing this specific grasp in the model. b) Kernel-based grasp density. The right-hand side shows lighter sampling for illustration purposes. D represents the density, while w_i and K_i represent the individual weights and kernels. c) Grasp success rates for the object 'basket' after different learning cycles (i) counting kinematic path planning errors as failures, and (ii) excluding such errors from the score. Red bars show the success rate of grasps before learning has been applied. Green bars correspond to grasps that have been drawn randomly from the learned grasp density. Blue bars correspond to the maximum-likelihood grasps from the learned grasp density. d) shows examples of a succeeding and a failing experiment. Figure adapted from [10], with kind permission by the authors.

orientation. Affordances are represented probabilistically with *grasp densities* [10], which correspond to continuous probability density functions defined on the 6D pose space. Their computational representation is non-parametric: A density is represented by a large number of weighted grasp observations. Density values are estimated by assigning a kernel function to each observation and summing the kernels [32]. An intuitive illustration of a grasp kernel is given in Fig. 6a and 6b illustrates a kernel-based grasp density.

OAC^{gen} is potentially applicable whenever the gripper is empty and an instance of object o is present in the scene. As in the previous example, the prediction function T always returns $\texttt{statusGrasp} = \texttt{stable}$. The attribute space $\mathcal{S}$ is defined by

$$\mathcal{S} = \{\mathrm{targetObj} = o, \texttt{statusGrasp}\}. \tag{4}$$

Here, the state description includes an attribute $\mathrm{targetObj}$ that specifies which object model o is to be applied by the `execute` function; this model is chosen by processes external to the OAC. M is defined in such a way as to maintain cumulative outcome statistics of executions of this OAC, updated via `updateM` (see Fig. 6c).

The `execute` function is defined in such a way as to return an experiment

$$\texttt{expr} = (s_o, \texttt{statusGrasp}_{t+1} = \texttt{stable}, \texttt{statusGrasp}_{t+1}),$$

in s_a, the attribute `statusGrasp` is the observed status after the

grasping attempt (see Fig. 6d). In addition, the data structures representing s_o, s_p and s_a may include further state information such as object and gripper poses. Such information is used, in particular, by `updateCP` to update the grasp density by integrating new experiments, which leads to increasingly reliable performance as can be seen in Fig. 6c.

5 Extraction of World knowledge by exploration

In this section we briefly describe the process of generating world knowledge by means of executing the generic and specific OACs. An important intermediate stage is sketched in Fig. 7. The top row in Fig. 7 represent the innate state of the system in which object and grasp memory is empty. It illustrates the usage of OAC^{gen} to grasp an unknown object based on the scene representation which is available in the iconic memory. Once an object is grasped, an object model is generated (see [17] for details). Subsequently the model can be used for pose-estimation in future scenes and thereby enable the association of actions to the specific object by OAC_o^{grasp} — this is illustrated in the bottom row in Fig. 7 representing a more advanced state of the system.

The experiments generated by the OACs coding object independent and object specific grasp knowledge are stored in the episodic buffer and are the basis for more abstracted representations in two respects. First, object dependent grasp knowledge is stored in the grasp densities (see section 5.2) and second, so called 'co-occurrence tables' store the statistics of feature relation - action associations (see section 5.3). Both kinds of knowledge are stored in the long term memory. Moreover, object shape knowledge is generated and also stored in the long term memory (see section 5.1).

5.1 Object shape knowledge

By successfully grasping a new rigid object initially (using OAC^{gen}) full physical control over it is achieved. This allows the object to be viewed from a variety of perspectives. From these views an accumulated description of multi-modal primitives is extracted. A detailed technical description of the accumulation process is given by Pugeault and Krüger, [28]. Besides other uses this generated shape description allows us to recognize the object in the scene and estimate its pose (see [11] for more information). The ability to estimate the object's pose is essential to be able to associate actions to the object as done by OAC_o^{grasp}.

5.2 Grasp Densities

The object specific grasp experiments generated by OAC_o^{grasp} are used to create grasp densities. The pose of each successful grasp defines one point in the 6-dimensional space and kernel density estimation is used to achieve a continuous density (see [10] and figure 6 for more details) based on the individual points. Once a grasp density is learned it can also be improved later on for instance by evaluating samples from it and use these to create a new, refined density which will lead to an improved success likelihood of OAC_o^{grasp}. Moreover, the grasp densities represent abstracted grasp knowledge which can be used for further learning, e.g., about how to improve the speed of convergence of the grasp densities (see section 6.1).

The width of each kernel is currently selected manually and depends on the specific use case. Typically they are chosen just large enough to ensure that neighbouring kernels overlap in order to ensure continuous density. The more detailed and fine grained the resulting density has to be, the more experiments are needed to ensure that the density is not incomplete and appears patchy. The level of detail depends one the usecase. When searching for a maxima it might be less critical how fine-grained the density is. For other investigations, e.g. as those described in section 5.3.1, it is beneficial to have a fine-grained density. As the density exists in a 6-dimensional space, the required number of experiments to reach a dense coverage grows quickly when a higher level of detail is required.

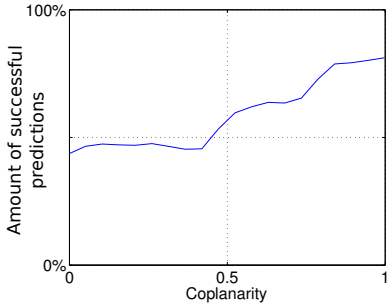


Figure 8: Using scenes containing one object, grasps have been created from pairs of contours using OAC^{gen} and compared against the grasps density associated with the object. This has been done using three different objects in total. The co-occurrence table shows the success likelihood of the grasps relative to the values of the feature-relations between the contours.

5.3 Co-occurrence tables

The experiments generated by OAC^{gen} can directly be used to improve the success likelihood of OAC^{gen} by calling the update function that is intrinsic to the OAC (illustrated in Fig. 3). Moreover, experiments generated by OAC_o^{grasp} are used to create so called co-occurrence tables (see Fig. 8 and 9) which represent projections of the grasp densities abstracted from the accumulated experiments. We discuss two different projections.

5.3.1 Co-occurrence tables for OAC^{gen}

The co-occurrence tables such as Fig. 8 store the values of relations between pairs of contours as well as the success likelihoods of grasps generated on these contour pairs and can thereby be used to improve the success likelihood of OAC^{gen}. Fig. 8 is based on grasps of three different objects and addresses coplanarity. It shows that coplanarity indeed seems to be an indicator for grasp affordances. Beside introducing additional relations, e.g. colinearity or cocolourity which is based on the colour-difference between those sides of the contours that face each other, also additional experiments using different objects will ensure that the statistic becomes more and more complete.

5.3.2 Co-occurrence tables for OAC_o^{grasp}

The kernels used in the grasp density approach in [10] are isotropic. This is unsatisfying in two respects. First, there is a certain arbitrariness in the selection of kernel parameters which requires manual selection. Second, it turns out that the convergence speed for the grasp densities is rather slow as many isotropic kernels are used to model the grasp affordances in sufficient detail. Both issues can be addressed when using the grasp densities as basis for the learning of the statistical dependencies of grasps in the vicinity of an already successfully tested grasp. More specifically, a simulation environment

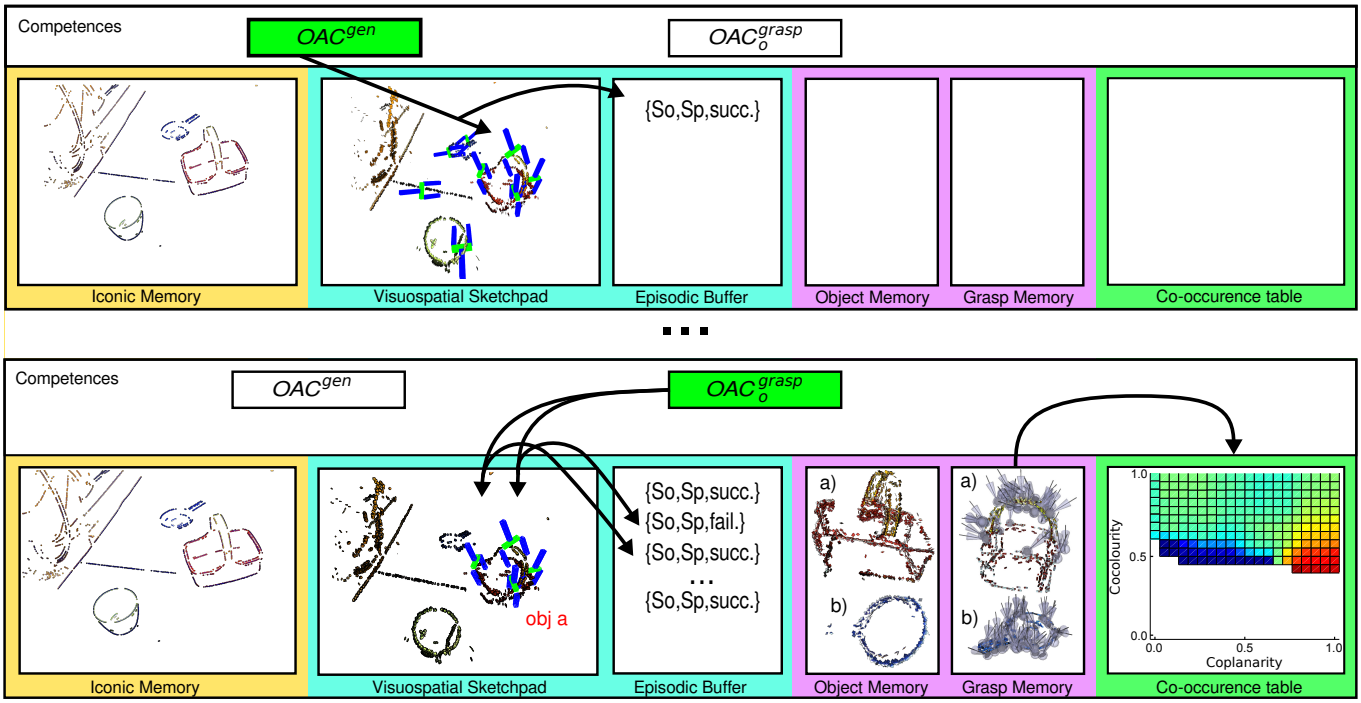
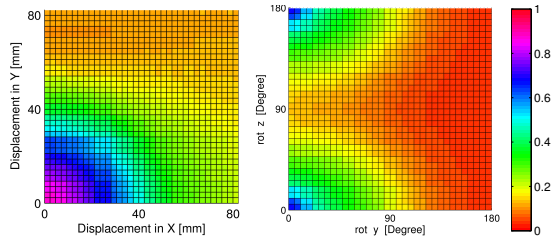


Figure 7: Illustration of how the system interacts by means of the two OACs interacting with the environment at two different stages of development. The top row represents this interaction at an innate state of development while the bottom row represents a more mature state.



(a) Translations (b) Rotations

Figure 9: Each successful grasp has been (a) translated or (b) rotated and subsequently its success likelihood is estimated using the grasp density associated with the object. This figure shows only the results for two dimensions of translation and rotation.

has been used to achieve a very dense grasp density, subsequently each successful grasp is transformed locally by a rigid body motion and using the density it is investigated whether the transformed action still would be successful. In order to reduce the complexity, only translations (see Fig. 9a) or rotations (see Fig. 9b) have been applied, not a combination of them. In these co-occurrence tables a clear anisotropy in the conditional probabilities are visible indicating that isotropic kernels are indeed a sub-optimal choice. In section 6.1 we argue that these co-occurrence tables can be used to define more optimal kernels.

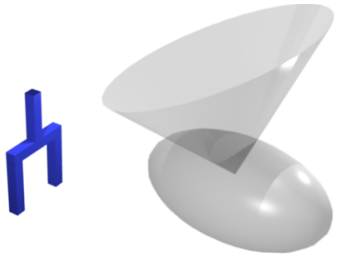
6 Interaction of the development of SMS and world knowledge

As indicated in Fig. 2, the SMSs and the generic world knowledge develop in parallel and complex interactions are to be expected. Making statements about this interactions in humans is difficult since only the change of behaviours, i.e., the executions of OACs/SMSs is directly observable. Statements of the change of internal representations are very difficult to achieve by means of neurophysiology. For example, it is virtually impossible to do single-cell recording experiments during development in awake behaving monkeys [23] (see also [16]). Developmental psychology can generate insights into that issue by means of sophisticated experiments. However, these insights are only indirect. Hence we find it to be valuable to look at such interactions in a developing robot system. This allows for making algorithmic problems explicit on a high level of detail.

In this section and based on the generic world knowledge accumulated by means of the OACs and abstracted in terms of grasp densities and co-occurrence tables as described in section 5, we intend to exemplify these interactions. First, we will discuss the need to improve the grasp density learning by means of learning more appropriate kernels in section 6.1. Then we discuss the role of co-occurrence tables for finding grasp affordances by means of statistics in section 6.2.

6.1 Kernel learning

The observations manifested in the co-occurrence tables in figure 9 lead to the idea of an anisotropic kernel where the iso-probable surface for the positional part becomes a ellipsoid rather than a sphere (see figure 10). We are currently working on developing these kernels in the density computation process. Besides having an empirical justification of the kernels themselves we also expect a much better convergence performance. The main benefit of the anisotropic kernel is that fewer kernels can be used to describe the density. As a direct consequence of this, we expect that fewer experiments are needed to achieve a “complete” density, which will then also speed up the convergence and will reduce the memory usage.



(a) Gripper (b) Kernel

Figure 10: (a) The orientation of the grasp corresponding to the mean-value of the kernel and (b) visualization of how an anisotropic kernel could be formed.

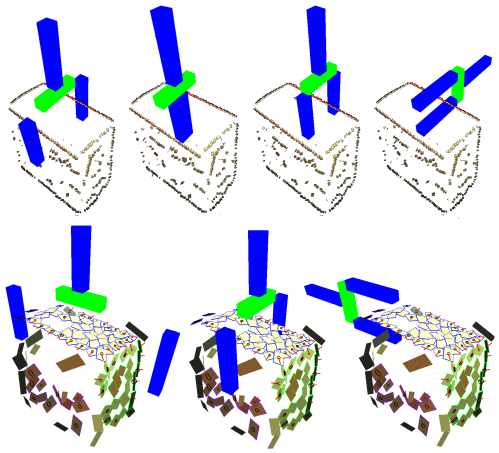


Figure 11: Top row: Grasping hypotheses derived from a pair of co-planar contours, (see section 4.2). Bottom row: Grasping hypotheses based on a single surface feature.

6.2 Co-occurrence tables as basis for justifying and improving grasp reflex

The co-occurrence tables also allow for a statistical justification of the originally hardwired behaviours as used in the execution of OAC^{gen}. Looking at the co-occurrence tables in Fig. 8 it becomes visible that the co-planarity relation is indeed indicative for successful grasps. It can be expected that the statistical analysis of the space of feature-relation action associations will reveal further indicative relations. In Fig. 11 besides edge pair related grasp affordances (top row) also surface related grasp affordances (bottom row) are shown. It is likely that many more indicative feature relation-grasp affordance relations do exist, potentially also for feature relations of very high order. Once enough grasp data in terms of grasp densities is available to the system even such higher order relations can be analysed. These can then be the basis for new OACs (i.e., branching OACs) coding more sophisticated grasp affordances.

Note that also in this context it is important to generate a large number of experiments and to integrate them in the co-occurrence tables. Hence the principle of ongoing learning on all levels as realised by the OACs is decisive for selecting the required material.

7 Related Work on Sensorimotor Schemas

Most work on explicitly Piaget-inspired sensorimotor schemas [12, 6, 35, 22] either does not have objects to perceive (e.g. using sensorimotor schemas to navigate in mazes), or has objects that are only sensed in a binary way (present or not), and so these are not comparable with our work on gathering knowledge of objects. On the other hand, recent work on affordances [37], though not explicitly modelling sensorimotor schemas, is quite close to our work. This work uses supervised learning to learn the patterns of visual features which are indicative of graspability (and other affordances). Given the appropriate training data, this approach can implicitly capture relationships which are similar to our co-occurrence tables for example. However our approach is capturing and representing object knowledge more explicitly, and this should give greater generality in application.

In an alternative approach, Hart and Grupen [14] describe a Piagetian inspired framework for constructing adaptive robot control strategies. While our OACs place little restrictions on control programs or learning schemes, Hart and Grupen's approach limits the usable control programs to a specific set of functions and the learning scheme to reinforcement learning. The authors show how the Piagetian notions of assimilation and accomodation are implemented an give examples of their usage within their system. Most interstingly they allow composition of schemas; this goes beyond the work reported here, and if combined with our ideas of developing generic world knowledge, this could potentially facilitate the learning of sensory abstractions which capture relations among objects, or spatial relationships (i.e. a higher order of world knowledge). This would be the next logical step for the learning of world knowledge in the upper track of Fig. 2, and according to Piageian theory it is through combinations of schemas that such knowledge is acquired [24, 25]).

8 Discussion

In this paper we demonstrated an important developmental process which is very hard to observe by means of developmental psychology or neurophysiology, namely the interaction of emerging generic world knowledge and developing sensory motor schemas. In this context, we have used the formalisation of sensory motor schemas in terms of object action complexes. We investigated two OACs coding grasping with and without prior object knowledge. Although still partly speculative, we could concretize potential interactions between developing generic world knowledge and the execution of OACs. In future work we will realise embodied systems in which such an interactive development will take place and we will further specify and quantify such interactions.

ACKNOWLEDGEMENTS

This work was supported by the EU Cognitive Systems project XPERIENCE (FP7-ICT-270273).

REFERENCES

[1] Karen E. Adolph, 'Learning to keep balance', *Advances in Child Development & Behavior*, **30**, 1–40, (2002).
[2] Emre Başeski, Nicolas Pugeault, Sinan Kalkan, Leon Bodenhagen, Justus H. Piater, and Norbert Krüger, 'Using Multi-Modal 3D Contours and Their Relations for Vision and Robotics', *Journal of Visual Communication and Image Representation*, **21**(8), 850–864, (2010).
[3] Alan D. Baddeley, *Essentials of Human Memory*, Psychology Press, Taylor and Francis, 1999.
[4] A. Bicchi and V. Kumar, 'Robotic Grasping and Contact: A Review', in *IEEE Int. Conf on Robotics and Automation*, pp. 348–353, (2000).
[5] Jeannette Bohg and Danica Kragic, 'Learning grasping points with shape context', *Robotics and Autonomous Systems*, (2010).

[6] H. H. Chaput, *The constructivist learning architecture: a model of cognitive development for robust autonomous robots*, Ph.D. dissertation, AI Laboratory, The University of Texas at Austin, 2004. Supervisors: Kuipers and Miikkulainen.

[7] Andy Clark and Annette Karmiloff-Smith, 'The cognizer's innards: A psychological and philosophical perspective on the development of thought', *Mind & Language*, **8**(4), 487–519, (1993).

[8] Jefferson A. Coelho, Jr., Justus H. Piater, and Roderic A. Grupen, 'Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot', *Robotics and Autonomous Systems*, **37**(2–3), 195–218, (2001).

[9] Fernando J. Corbacho and Michael A. Arbib, 'Schema-based learning: Towards a theory of organization for biologically-inspired autonomous agents', in *Agents*, pp. 520–521, (1997).

[10] R. Detry, D. Kraft, A. G. Buch, N. Krüger, and J. Piater, 'Refining grasp affordance models by experience'. International Conference on Robotics and Automation, 2010.

[11] R. Detry, N. Pugeault, and J. Piater, 'A probabilistic framework for 3D visual object representation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**(10), 1790–1803, (2009).

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

[13] Kurt W. Fischer, 'A theory of cognitive development: The control and construction of hierarchies of skills', *Psychological Review*, **87**(6), 477–531, (1980).

[14] Stephen Hart and Roderic Grupen, 'Learning generalizable control programs', *IEEE Transactions on Autonomous Mental Development (Accepted for publication)*, (2010).

[15] K. Huebner, S. Ruthotto, and D. Kragic, 'Minimum volume bounding box decomposition for shape approximation in robot grasping', in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1628 –1633, (May 2008).

[16] Z. Kourtzi, M. Augath, NK. Logothetis NK, JA Movshon, and L. Kiorpes, 'Development of visually evoked cortical activity in infant macaque monkeys studied longitudinally with fmri', *Magn Reson Imaging*, **24**(4), 359–66, (2006).

[17] Dirk Kraft, Renaud Detry, Nicolas Pugeault, Emre Başeski, Frank Guerin, Justus Piater, and Norbert Krüger, 'Development of object and grasping knowledge by robot exploration', *IEEE Transactions on Autonomous Mental Development*, **2**(4), 368–383, (2010).

[18] Norbert Krüger, Dirk Kraft, Justus Piater, Florentin Wörgötter, Alejandro Agostino, Aleš Ude, Damir Omrčen, Mark Steedman, Ron Petrick, Christopher Geib, Bernhard Hommel, Danica Kragic, Jan-Olof Eklundh, Volker Krüger, Tamim Asfour, and Rüdiger Dillmann, 'A formal definition of object action complexes and examples at different levels of the process hierarchy', *PACOplus report, http://wwwiaim.ira.uka.de/pacoplus/download/OAC-Definition.pdf*, (2009).

[19] Jeffrey J. Lockman, 'A perception-action perspective on tool use development', *Child Development*, **71**(1), 137–144, (2000).

[20] A.T. Miller, S. Knoop, H.I. Christensen, and P.K. Allen, 'Automatic grasp planning using shape primitives', *ICRA*, (2003).

[21] Joseph Modayil and Ben Kuipers, 'Autonomous development of a grounded object ontology by a learning robot', in *Proceedings of the AAAI Spring Symposium on Control Mechanisms for Spatial Knowledge Processing in Cognitive/Intelligent Systems*. AAAI, (2007).

[22] F.S. Perotto, J.C. Buisson, and L.O. Alvares, 'Constructivist anticipatory learning mechanism (CALM): Dealing with partially deterministic and partially observable environments', in *Proceedings of the Seventh International Conference on Epigenetic Robotics, Piscataway, NJ, USA*, pp. 117–127, (2007).

[23] Peter Jannsen (personal communication), (2011).

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

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

[26] J. Piaget, *The psychology of intelligence*, 1976.

[27] Mila Popović, Dirk Kraft, Leon Bodenhagen, Emre Başeski, Nicolas Pugeault, Danica Kragic, Tamim Asfour, and Norbert Krüger, 'A strategy for grasping unknown objects based on co-planarity and colour information', *Robotics and Autonomous Systems*, **58**(5), 551 – 565, (2010).

[28] N. Pugeault and N. Krüger, 'Temporal accumulation of oriented visual features', *Journal of Visual Communication and Image Representation*, (in press).

[29] N. Pugeault, F. Wörgötter, and N. Krüger, 'Visual primitives: Local, condensed, and semantically rich visual descriptors and their applications in robotics', *International Journal of Humanoid Robotics (Special Issue on Cognitive Humanoid Vision)*, **7**(3), 379–405, (2010).

[30] Ashutosh Saxena, Lawson L. S. Wong, and Andrew Y. Ng, 'Learning grasp strategies with partial shape information', in *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, pp. 1491–1494. AAAI Press, (2008).

[31] A. Sheya and L. B. Smith, 'Development through sensory-motor coordinations', in *Festchrift for F. Varela*, MIT Press, (2010).

[32] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall/CRC, 1986.

[33] L. B. Smith, 'Dynamic systems, sensori-motor processes and the origins of stability and flexibility', in *Toward a unified theory of development: Connectionism and dynamic systems theories re-considered*, eds., J. Spencer, M. Thomas, and J. McClelland, Oxford University Press, (1997).

[34] Ad W. Smitsman and Ralf F. A. Cox, 'Perseveration in tool use: A window for understanding the dynamics of the action-selection process', *Infancy*, **13**(3), 249–269, (2008).

[35] Georgi Stojanov, 'Petitagé: A case study in developmental robotics', in *Proceedings of Epigenetic Robotics 1*, eds., C. Balkenius, J. Zlatev, H. Kozima, K. Dautenhahn, and C. Breazeal, (2001).

[36] E. Thelen and V. Whitmyer, 'Using dynamic field theory to conceptualize the interface of perception, cognition and action', in *The Minnesota Symposium on child psychology: Vol. 33. Action as an organizer of learning and development*, eds., J. J. Rieser, J. J. Lockman, and C. A. Nelson, Mahwah, NJ: Lawrence Erlbaum Associates, Inc., (2005).

[37] E. Ugur, E. Sahin, and E. Oztop, 'Affordance learning from range data for multi-step planning', in *Ninth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems (EpiRob)*, pp. 177–184, (2009).

[38] F. Wörgötter, A. Agostini, N. Krüger, N. Shyloa, and B. Porr, 'Cognitive agents — a procedural perspective relying on the predictability of object-action-complexes (OACs)', *Robotics and Autonomous Systems*, **57**(4), 420–432, (2009).

Learning STRIPS operators from noisy and incomplete observations*

Kira Mourão
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
kmourao@inf.ed.ac.uk

Luke Zettlemoyer
Computer Science & Engineering
University of Washington
Seattle, WA98195
lsz@cs.washington.edu

Ronald P. A. Petrick
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
rpetrick@inf.ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

When agents learn to act autonomously in real-world domains, they must acquire a model of the dynamics of the domain in which they operate. However, learning domain dynamics can be challenging, especially where an agent only has partial access to the world state, or external sensors that generate noise. Even in standard STRIPS domains, existing approaches cannot learn from noisy, incomplete observations typical of real-world domains. Here we propose a method which learns STRIPS action models in such domains, by decomposing the problem into first learning a transition function between states in the form of a set of classifiers, and second deriving explicit STRIPS rules from the classifiers' parameters. We evaluate our approach by learning action models from simulated standard planning domains taken from the International Planning Competition, and comparing the learnt models with the original domain models. Our results show that the approach can learn useful domain descriptions despite the presence of noise and partial observability.

Introduction

Developing agents with the ability to act autonomously in the world is one of the goals of artificial intelligence. One important aspect of this development is the acquisition of domain models to support planning and decision-making: to operate effectively in the world, an agent must be able to accurately predict when its actions will succeed, and what effects its actions will have. Only when a reliable action model is acquired can the agent usefully combine sequences of actions into plans, in order to achieve wider goals. However, learning domain dynamics can be a challenging problem: agents' observations may be noisy, or incomplete; actions may be non-deterministic; the world may be noisy or contain many irrelevant objects and relations.

In this paper we consider the problem of acquiring explicit domain models from the raw experiences of an agent exploring the world, where the agent's observations are incomplete, and observations and actions are subject to noise. The domains we consider are relational STRIPS (Fikes and Nilsson 1971) domains, although our approach has the potential to be extended to more expressive domains. Furthermore, given the autonomous learning setting we restrict previous domain knowledge to that which could realistically have been acquired autonomously by the agent. Thus we assume a weak domain model where the agent knows how to identify objects, has acquired predicates to describe object attributes and relations, and knows what types of actions it may perform, but not the appropriate contexts for the actions, or their effects. Experience in the world is then developed through observing changes to object attributes and relations when motor-babbling with primitive actions.

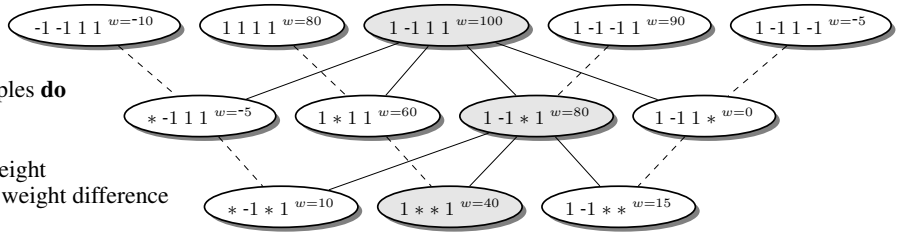
Previous work fulfils some but not all of the requirements for learning models in the setting we consider. In autonomous robotics, various techniques exist to learn preconditions and effects of actions in noisy, partially observable worlds (Doğar et al. 2007; Metta and Fitzpatrick 2003; Modayil and Kuipers 2008). However, none of these approaches learn relational models. Conversely, much previous work on learning relational action models relies on the provision of prior knowledge of the action model. Strategies include seeding initial models with approximate planning operators (Gil 1994), using known successful plans (Wang 1995), excluding action failures (Amir and Chang 2008), or the presence of a teacher (Benson 1996). Such knowledge is unlikely to be available to an autonomous agent learning the dynamics of its domain. Additionally, few approaches are capable of learning under either partial observability (Amir and Chang 2008; Yang, Wu, and Jiang 2007; Zhuo et al. 2010), noise in any form (Pasula, Zettlemoyer, and Kaelbling 2007; Rodrigues, Gérard, and Rouveirol 2010), or both (Halbritter and Geibel 2007; Mourão, Petrick, and Steedman 2010). Approaches which handle both do so by generating implicit action models which must be used as a black-box to make predictions of state changes, and do not generate symbolic action representations.

We tackle the problem of learning STRIPS operators from noisy and incomplete observations by developing a two-stage approach. Our motivation for this approach is to decouple the requirement to tolerate noisy, incomplete observations from the requirement to learn compact STRIPS operators. The advantage of this decoupling is that the initial action model acts as a noise-free, fully observable source of observations from which to extract explicit action rules. In the first stage we learn action models by constructing a set of kernel classifiers which tolerate noise and partial observability, but whose action models are implicit in the learnt parameters of the classifiers, similar to the work of Mourão,

*Submitted to AAAI 2012

for $v \in SV^+$ **do**
  $child := v$
  while $child$ only covers +ve training examples **do**
    $parent := child$
    for each valued bit in $parent$ **do**
      flip bit to its negation and calculate weight
    $child :=$ child whose parents have least weight difference
  $rule_v := parent$

(a) Rule extraction algorithm



(b) Example of rule extraction process

Figure 1: Each node in (b) contains a vector corresponding to a possible precondition, and the weight w assigned to the vector by the voted perceptron model. Each level of the lattice contains vectors with one fewer feature than the level above. Lines join parent and children nodes: solid lines link the candidate parent rule at one level with its children in the level below, and dashed lines link children to their alternative parent. Shaded nodes are the preconditions selected at each iteration through the lattice. The positive support vector "seed" is the vector $\langle 1$ -1 1 $1\rangle$ with weight 100. Following the rule extraction algorithm in (a), the child whose parents have the least weight difference, the vector $\langle 1$ -1 * $1\rangle$, is chosen as the next candidate rule. The process ends with the rule $\langle 1$ * * $1\rangle$ as both children have a negative counterexample in the training data (not shown).

Petrick, and Steedman (2009; 2010). However, we additionally use the method to learn preconditions as well as effects, as suggested but not explored in earlier work. Also, we evaluate additional kernels for the learning problem and select a better performing alternative. In the second stage we devise a novel method to derive explicit STRIPS operators from the model implicit in the kernel classifiers. In experiments the resulting rules perform as well as the original classifiers, while also providing a compact representation of the action models suitable for use in automated planning systems.

The Learning Problem

A *domain* $\mathcal{D}$ is defined as a tuple $\mathcal{D} = \langle \mathcal{O}, \mathcal{P}, \mathcal{A} \rangle$, where $\mathcal{O}$ is a finite set of world objects, $\mathcal{P}$ is a finite set of predicate (relation) symbols, and $\mathcal{A}$ is a finite set of actions. Each predicate and action also has an associated arity. A *fluent expression* is a statement of the form $p(c_1, c_2, ..., c_n)$, where $p \in \mathcal{P}$, n is the arity of p, and each $c_i \in \mathcal{O}$. A *state* is any set of fluent expressions, and $\mathcal{S}$ is the set of all possible states. For any state $s \in \mathcal{S}$, a fluent expression ϕ is true at s iff $\phi \in s$. The negation of a fluent expression, $\neg\phi$, is true at s (also, ϕ is false at s) iff $\phi \notin s$.

Each action $a \in \mathcal{A}$ is defined by a set of *preconditions*, Pre_a, and a set of *effects*, Eff_a. Pre_a can be any set of fluent expressions and negated fluent expressions. Each $e \in Eff_a$ has the form $add(\phi)$ or $del(\phi)$, and ϕ is any fluent expression. Objects mentioned in the preconditions or the effects must be listed in the action parameters (the *STRIPS scope assumption* (SSA) (Walsh and Littman 2008)). Action preconditions and effects can also be parameterised. An action with all of its parameters replaced with objects from $\mathcal{O}$ is said to be an *action instance*.

The task of the learning mechanism is to learn the associations between action-precondition pairs and their effects, that is, rules of the form $\langle A, Pre_A \rangle \rightarrow Eff_A$. The mechanism learns from sequences of interleaved state observations and actions, including action failures.

Learning implicit action models

The basis of our approach is the division of the learning problem into two parts: initially a classification method is used to learn an implicit action model, then explicit rules are derived from the resulting action representations. In learning an implicit action model, our approach follows earlier work (Croonenborghs et al. 2007; Halbritter and Geibel 2007; Mourão, Petrick, and Steedman 2009; 2010) which encodes the learning problem in terms of the inputs and outputs of a set of classifiers. However, none of these methods generate explicit rules describing the learnt action models.

We only briefly describe our approach to learning implicit action models below, as it is an application of previous work (Mourão, Petrick, and Steedman 2009; 2010), with minor modifications. We first reduce the size of the state descriptions by only considering objects which are mentioned in the action parameter list. By the SSA this is sufficient to learn STRIPS rules. For an action instance a with arguments $o_1, o_2, ..., o_n$ we therefore restrict the state description to all possible fluents whose arguments are in $\{o_1, o_2, ..., o_n\}$. Additionally we schematise the descriptions by replacing the i-th action parameter with the label arg_i whenever it appears in any fluent. Thus all state descriptions are now written in terms of the action parameters rather than in terms of any specific objects. The SSA fixes a small number of objects to consider for an action, as well as their roles, which allows relational state descriptions to be encoded in a vector, as each possible fluent in a state maps to exactly one possible fluent in any other state. We then encode state descriptions as vectors where each bit in the vector corresponds to each possible fluent which could exist in the schematised state description. The value of a bit is 1 if the fluent is true, -1 if false, and a wildcard value * if the fluent is unobserved.

A set of classifiers now learns the implicit action model, where each classifier takes as input an action and a state description vector, and predicts change to a single bit of the state description. We use voted perceptron classifiers combined with a DNF kernel, $K(x, y) = 2^{same(x,y)}$, where

$same(x, y)$ is the number of bits with the same value in both x and y (Sadohara 2001; Khardon and Servedio 2005). In the $same(x, y)$ calculation, bits with unobserved values are excluded. The DNF kernel has features which are all possible conjunctions of fluents, ideal for learning where the true underlying hypothesis is expected to be a conjunction of fluents, as for action preconditions. However, DNF is not PAC-learnable by a perceptron using the DNF kernel, as there exist examples on which it can make exponentially many mistakes (Khardon, Roth, and Servedio 2005). We therefore also consider the k-DNF kernel, whose features are all possible conjunctions of fluents of length $\leq k$ for some fixed k: $K(x, y) = \sum_{l=0}^{k} \binom{same(x,y)}{l}$ (Khardon and Servedio 2005). Preconditions with more than k fluents are still possible since the voted perceptron supports hypotheses which are conjunctions of features.

Rule extraction

The first step in deriving explicit action rules is to extract individual rules to predict each fluent in isolation. We will look at how to combine the rules to make full *add*(ϕ) and *del*(ϕ) lists in the next section. Rules are extracted from a voted perceptron with kernel K and support vectors $SV = SV^+ \cup SV^-$, where SV^+ (SV^-) is the set of support vectors whose *predicted* values are 1 (-1). The positive support vectors are each instances of some rule learnt by the perceptron, and so are used to "seed" the search for rules. The extraction process aims to identify and remove all irrelevant bits in each support vector, using the voted perceptron's prediction calculation to determine which bits to remove.

The *weight* of any possible state description vector $\mathbf{x}$ is defined to be the value calculated by the voted perceptron's prediction calculation before thresholding (Freund and Schapire 1999). The predicted value for $\mathbf{x}$ is 1 if $weight(\mathbf{x}) > 0$ and -1 otherwise. A *child* of vector $\mathbf{x}$ is any distinct vector obtained by replacing a single bit of x with the value *. Similarly, a *parent* of $\mathbf{x}$ is any vector obtained by replacing a *-valued bit with the value 1 or -1.

The basic intuition behind the rule extraction process is that more discriminative features will contribute more to the weight of an example. By repeatedly deleting the feature which contributes least to the weight, we should be left with the most discriminative features underlying an example, which can be used to form a precondition. An example of the process of extracting rules is shown in Figure 1(b), and an outline of the algorithm in Figure 1(a), as follows. Take each positive support vector v in turn, and aim to find a conjunction $rule_v$ which covers v and does not cover any negative training examples, but where every child of $rule_v$ covers at least one negative example. Construct $rule_v$ by a greedy algorithm which first takes v as a candidate rule and then repeatedly selects a new candidate rule by taking the child of the current candidate whose parents have the least difference in weight – that is, for $\mathbf{x} = (x_1, ..., x_i, ..., x_n)$ and $\mathbf{x}_{\neg i} = (x_1, ..., \neg x_i, ..., x_n)$, finding

$$\underset{x_i \in \{x_1,...,x_i,...,x_n\}}{\operatorname{argmin}} (weight(\mathbf{x}) - weight(\mathbf{x}_{\neg i})).$$

Removing the resulting x_i removes the least discriminative bit in the current candidate rule. At each step the new candidate rule is tested against the training examples. If it classifies a negative training example as positive, then the rule is too general and $rule_v$ is set to the previous candidate rule, otherwise the process repeats. The result is a set of rules for each action, predicting when a particular output bit changes. There may be many rules, up to one per positive support vector, each consisting of a set of preconditions which, if satisfied, predict the output bit will change.

$allRules :=$ rules from rule extraction process
$hwr :=$ highest weighted rule in $allRules$
$baseline := (hwr.pre, \varnothing)$
$locks = \varnothing$
while $allRules \neq \varnothing$ **do**
  $next :=$ highest weighted rule in $allRules$
  $allRules := allRules \setminus \{next\}$
  if $CompatibleEffects(baseline, next)$ **then**
    if $CombinePrecons(baseline, next, newpre, locks)$ **then**
      $BacktrackPrecons(baseline, next, newpre)$
      if $AcceptPrecons(baseline, next, newpre)$ **then**
        $baseline.pre := newpre$
      if $AcceptEffect(baseline, next.eff)$ **then**
        $baseline.eff := baseline.eff \cup next.eff$
      $BacktrackEffects(baseline)$

Figure 2: Outline rule combination algorithm

Rule combination

The rule combination process aims to build a single conjunctive rule for each action. It operates on all rules ($\langle precondition, effect \rangle$ pairs) for an action produced by the rule extraction process. For each action it first initialises the baseline rule to a default rule consisting of the precondition of the highest weighted rule, and no effects. The baseline rule is then refined by attempting to combine it with each of the remaining pairs in turn, in order of highest weight. Each time the process must generate a suitable candidate rule, and then decide whether to accept or reject the candidate as the new baseline (Figure 2). Without noise or partial observability, the combination process is a straightforward conjunction of all preconditions and all effects in the set of rules for an action. Additional checks and backtracking are needed to identify and eliminate fluents introduced by noise while adding in fluents omitted due to partial observability.

In attempting to combine the current baseline rule with a new rule, the first check is whether the new rule's effect contradicts any effect in the baseline rule ($CompatibleEffects$). Effects conflict if both rules predict change to a fluent, but the rules have different values for the fluent in their preconditions. If there is a conflict, the new rule is rejected, as we assume only one rule per action, and the higher weighted baseline rule is more likely to be correct.

Rules can be combined on both effects and preconditions, and each of these is considered separately, since rejecting a noisy precondition does not necessarily mean that the corresponding effect should also be rejected. First, an attempt is made ($CombinePrecons$) to combine the rule precondi-

tions with the baseline preconditions, to form a set of candidate preconditions $newpre$. If the preconditions of the two rules do not contradict, they can be combined by conjunction to form the candidate preconditions. Otherwise, the non-conflicting fluents are conjoined, and an attempt is made to reconcile the remaining conflicts.

For each conflicting fluent, there are three possible values the fluent could take in the preconditions of the true rule: $*$ (unobserved), 1 (true) or -1 (false). The weight $weight_{\mathit{eff}}$ (for each effect *eff* in the baseline effects) of each variant is calculated. The preferred variant is where the value is $*$, indicating a non-discriminative feature, and giving the simplest precondition. However, a variant is only acceptable if the weight of the resulting rule is positive for all of its effects, since then it still predicts the same effects as the baseline rule. If accepted, the fluent is locked at the $*$ value, to prevent later, possibly noisy rules, from resetting it. If the $*$-variant is unacceptable, then the (1)-valued or (-1)-valued cases are considered, provided they have positive weights on all the effects. If both variants are acceptable, whichever has the highest average weight over all the effects is selected. If neither variant is acceptable then the conflict is unresolved for this fluent. As long as the conflicts on every fluent are resolved, the rule combination process can continue with the new candidate rule. If not, the current rule is rejected.

$CombinePrecons$ allows many additional preconditions to be added untested to the candidate, and so at this stage, a backtracking step ($BacktrackPrecons$) generates a number of alternative, less specific preconditions from the candidate preconditions. For each fluent in the candidate which does not exist in the original preconditions, an alternative candidate precondition is constructed, without that fluent. If the scoring function ($AcceptPrecons$) rates the baseline precondition as worse than the alternative precondition, then the fluent is removed from the candidate preconditions.

Next, the resulting candidate rule is compared with the baseline rule, using a separate scoring function for the preconditions ($AcceptPrecons$) and effects ($AcceptEffects$). If the precondition scoring function rates the candidate preconditions as acceptable, they become the new baseline preconditions, and similarly for the effects. Candidate preconditions may be accepted without the effects, and vice versa.

The precondition scoring function ($AcceptPrecons$) accepts the candidate preconditions if they are no worse than the existing preconditions, when used to predict any of the effects in the candidate rule. It makes use of both the weight calculation and coverage of the training examples. Both should be considered, as weight alone may permit rules which do not cover any training examples, while coverage alone may allow negatively weighted rules. Thus a combination of two rules can be rejected if, for any of the resulting effects e, the resulting precondition p does not have a positive weight: $weight_e(p) \leq 0$, or if it does not cover any training example: $coverage_e(p) = 0$. Coverage is defined to account for partial observability, so that precondition p covers example x at effect e ($covers_e(p, x)$) if none of the fluents in the example state contradict the fluents in the rule preconditions, and e is in both the example state changes and the rule effects. The coverage of p on the training set at effect e is defined to be the number of training examples p covers: $coverage_e(p) = |\{x : covers_e(p, x)\}|$.

Additionally, the precondition scoring function uses differences in precision and recall to identify and reject any rule which performs significantly worse than its comparison rule. It rejects rules where either the precision or recall drops substantially for any of the rules' effect bits. Since precision and recall is a trade-off, the comparison is made using the F-score for precondition pre at effect *eff*: $F_{pre,\mathit{eff}}$. If the new F-score $F_{new,\mathit{eff}}$ is less than ϵ_p times the baseline F-score $F_{baseline,\mathit{eff}}$, on any effect *eff*, then the new rule is rejected. In the results presented here, ϵ_p is set to 0.95.

The effects scoring function ($AcceptEffect$) similarly compares F-scores. It takes a single precondition and compares the F-score of one selected effect against the F-score for every other effect. This identifies effects which are inconsistent with the other effects in terms of precision and recall. In particular, effects which occur in far fewer examples than other effects are identified in this way: these are likely to be caused by noise, or could be conditional effects. An effect is rejected by the function if its F-score is less than ϵ_e times the F-score on any other effect of the same rule. In the results presented here, ϵ_e is set to 0.5.

Finally, the effects backtracking function ($BacktrackEffects$) tests if any of the effects should be removed from the rule, in light of the new preconditions. For instance, more specific preconditions may lower the incidence of some effects (as seen in the training data) to the extent that $AcceptEffects$ rejects them. Each effect is tested by the effects scoring function and, if rejected, removed from the rule's effects.

At this point, the preconditions and effects are written as vectors. The conversion from a vector precondition to a STRIPS precondition is straightforward: given the action and its parameters, the fluent and parameters corresponding to a particular bit position are known, and the value of the bit gives the value of the fluent. In the case of the effects, the bits only indicate whether a particular fluent changes, but not what the change is from or to. If the fluent also exists in the preconditions then this can be used to determine the correct value of the fluent in the effects; otherwise, the value can be obtained from the support vector from which the rule for the effect bit originated.

Experiments

We tested our approach on several simulated domains taken from the International Planning Competition (IPC) at `http://ipc.icaps-conference.org/`. The domains differ in terms of the number and arity of actions and predicates, and the number and hierarchy of types. The main domain characteristics are detailed in Table 1.

Sequences of random actions and resulting states were generated from the PDDL domain descriptions and used as training and testing data. All data was generated using the Random Action Generator 0.5 available at `http://magma.cs.uiuc.edu/filter/`, modified to additionally generate action failures. Table 2 shows the numbers of objects used in training and testing data for each domain.

Domain	Actions		Predicates	
	No.	Max arity	No. (+types)	Max arity
BlocksWorld	4	2	5	2
Depots	5	4	6 (+6)	2
ZenoTravel	5	6	8 (+4)	2
DriverLog	6	4	6	2
Rovers	9	6	25 (+8)	3

Table 1: Domain characteristics

Domain	Training	Testing
BlocksWorld	13 blocks	30 blocks
Depots	1 depot	4 depots
	2 distributors	4 distributors
	2 trucks	4 trucks
	3 pallets	10 pallets
	3 hoists	8 hoists
	10 crates	8 crates
ZenoTravel	5 cities	10 cities
	3 planes	5 planes
	7 people	10 people
DriverLog	3 road junctions	20 road junctions
	3 drivers	5 drivers
	7 packages	25 packages
	3 trucks	5 trucks
Rovers	2 rovers	4 rovers
	4 waypoints	8 waypoints
	3 objectives	4 objectives
	3 cameras	4 cameras
	3 objectives	4 objectives
	3 modes	3 modes
	2 stores	4 stores
	1 lander	1 lander

Table 2: Numbers of objects in training and testing worlds

Ten different randomly generated training and testing sets were used. Each training set was a sequence of 20,000 actions, and each testing set a sequence of 2,000 actions. Both sequences contained an equal mixture of successful and unsuccessful actions (where some precondition of the action was not satisfied, and so no change occurred in the world). In some domains (e.g. Rovers), portions of the state space can only be traversed once, and in these cases multiple shorter sequences of 400 actions were generated from randomly generated starting states.

In line with previous work (Amir and Chang 2008), incomplete observations were simulated by randomly selecting a fraction (10%, 25% or 50%) of fluents (including negations) from the world to observe after each action. The remaining fluents were discarded and the reduced state vector was generated from the observed fluents. Sensor noise was simulated similarly by flipping the value of each bit in the state vector with probability 1% and 5%.

Results

We first tested the performance of different kernels on learning the implicit action model, comparing results for a standard (non-kernelised) perceptron, a voted (non-kernelised) perceptron, and a voted kernel perceptron. Both the DNF kernel and k-DNF kernel with $k = 2, 3$ and 5 were tested. Performance was measured in terms of the F-score of the predictions on the test sets.

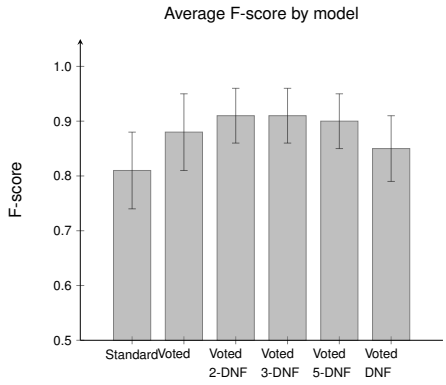


Figure 3: Comparison of the performance of different perceptrons learning action models from 20,000 random actions in STRIPS domains, averaged across all domains, levels of noise and partial observability. Error bars are standard error. Performance is significantly different between models which use a k-DNF kernel and those which do not.

The fully observable, noiseless cases are easily learnt by any of the perceptrons tested. After 5,000 training examples, the F-score on the test set is 1, in almost all cases. Performance of the voted perceptron, with or without the various kernels, is almost identical (results not shown). With the introduction of unobserved fluents or noise, the voted perceptron performs better than the standard perceptron. However, the DNF kernel does not improve performance, with the unkernelised voted perceptron learning significantly more accurate action models. In contrast, the k-DNF kernels all produce significantly more accurate models than the DNF kernel or no kernel ($p < 0.05$, repeated measures ANOVA with post-hoc Bonferroni t-test). Figure 3 gives a comparison of the relative performance of each model. The use of k-DNF kernels therefore represents a significant improvement on previous work which used only the DNF kernel. In light of these results, the 3-DNF kernel was selected for the remainder of the experiments.

Next, we extracted explicit rules from the implicit action models. There was no statistically significant difference between the F-scores of predictions made by the perceptron models and those made by the extracted rules (repeated measures ANOVA, $p > 0.05$). We also compared the resulting models to the original domain descriptions using a measure of error rate (Zhuo et al. 2010). The error rate for a single action is defined as the number of extra or missing fluents in the preconditions and effects (E_{pre} and E_{eff} respectively) divided by the number of possible fluents in the preconditions and effects (T): $Error(a) = \frac{1}{2}\left(\frac{E_{pre}+E_{\mathit{eff}}}{T}\right)$. The error rate of a domain model with a set of actions A is: $Error(A) = \frac{1}{|A|}\sum_{a\in A} Error(a)$.

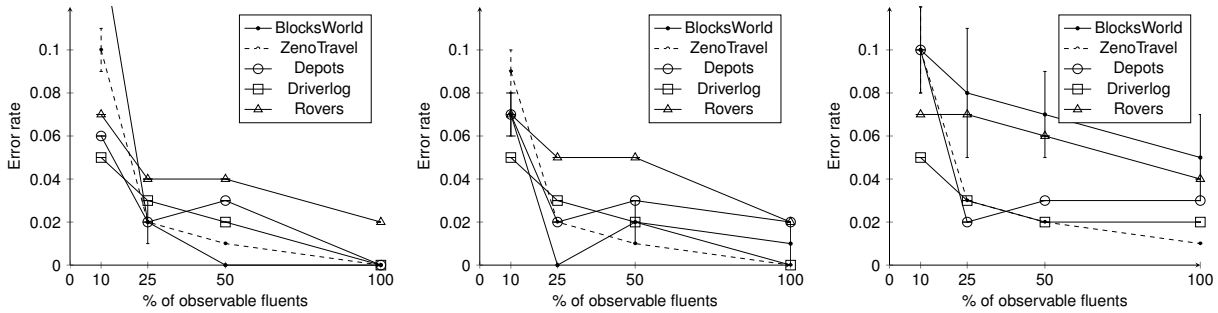


Figure 4: Results from learning explicit action rules from 5,000 training examples at varying levels of observability and noise in simulated planning domains. The error rate measures errors in the learnt domain model relative to the actual domain model.

```
(:action DEBARK
:parameters (?x1 ?x2 ?x3 )
:precondition (AND (in ?x1 ?x2) (at ?x2 ?x3))
:effect (AND (at ?x1 ?x3) (NOT(in ?x1 ?x2))))

(:action BOARD
:parameters (?x1 ?x2 ?x3 )
:precondition (AND (at ?x1 ?x3) (at ?x2 ?x3))
:effect (AND (NOT(at ?x1 ?x3)) (in ?x1 ?x2)))

(:action FLY
:parameters (?x1 ?x2 ?x3 ?x4 ?x5 )
:precondition (AND (at ?x1 ?x2) (fuel-level ?x1 ?x4)
        (next ?x5 ?x4))
:effect (AND (NOT(at ?x1 ?x2)) (at ?x1 ?x3)
        (NOT(fuel-level ?x1 ?x4)) (fuel-level ?x1 ?x5)))

(:action ZOOM
:parameters (?x1 ?x2 ?x3 ?x4 ?x5 ?x6 )
:precondition (AND (at ?x1 ?x2) (fuel-level ?x1 ?x4)
        (next ?x6 ?x5) (next ?x5 ?x4))
:effect (AND (NOT(at ?x1 ?x2)) (at ?x1 ?x3)
        (NOT(fuel-level ?x1 ?x4)) (fuel-level ?x1 ?x6)))

(:action REFUEL
:parameters (?x1 ?x2 ?x3 ?x4 )
:precondition (AND (fuel-level ?x1 ?x3) (next ?x4 ?x3)
        (next ?x3 ?x4) (at ?x1 ?x2))
:effect (AND (NOT(fuel-level ?x1 ?x3)) (fuel-level ?x1 ?x4))
```

Figure 5: Explicit action model output for the ZenoTravel domain after 5,000 training examples with 10% observability and 5% noise. Missing fluents are in bold italic, incorrect fluents in bold. The error rate of this example is 0.05.

The error rates indicate that the learnt models are close to the actual STRIPS domain definitions, falling below 0.1 after around 5,000 examples in all cases (Figure 4). In particular, for fully observable, noiseless domains the correct STRIPS model is given by the extracted rules in fewer than 2,000 training examples, except for the most complex domain, Rovers. Comparisons with other approaches in the literature are difficult due to differences in the learning settings. Nevertheless it is notable that the error rates of the learnt action models are low in comparison to action models learnt by Yang, Wu, and Jiang (2007) for the same domains: their error rates at 90% observability (the highest reported) range from around 0.04 (ZenoTravel) to 0.1 or above (DriverLog and Depots) to more than 0.6 (Rovers). An example action model is shown in Figure 5, demonstrating that the method derives compact STRIPS-like rules even with high levels of incompleteness and noise in the observations.

Conclusions and Future Work

The results demonstrate that our approach successfully learns STRIPS operators from noisy, incomplete observations, in contrast to previous work which either generates explicit operators but cannot tolerate noise and incomplete examples, or tolerates noise and incomplete examples but does not generate explicit operators. We also show empirically that the 3-DNF kernel is a more appropriate choice than the DNF kernel for learning in this setting.

Our approach depends on decomposing the learning problem into two stages: learning implicit action models and then deriving explicit rules from the implicit models. Crucially, the implicit models produce noise-free, complete observations *for the domain model which has been learnt*. An alternative approach to our rule derivation process would be to apply existing action model learning techniques to the observations produced by the implicit models. However such an approach effectively restarts the learning process, ignoring information already learnt and available in the perceptron models, and so is likely to be less efficient.

Our approach also depends on the STRIPS scope assumption (SSA) which essentially identifies the objects which are relevant to the action and fixes their roles. In real-world scenarios the SSA may not apply. Without the SSA, during learning we must also consider state relating to objects which are not listed in the action parameters. Implicit action models in this setting may be learnt using a graphical representation of states combined with a suitable graph kernel (Mourão 2012). In future work we therefore plan to extend our rule extraction method to derive rules from classifiers trained with graphical state representations. Additional steps will be required to efficiently handle the complexity introduced by the requirement to perform comparisons between graphical state descriptions.

Acknowledgements

This work was partially funded by the European Commission through the EU Cognitive Systems project Xperience (FP7-ICT-270273) and the UK EPSRC/MRC through the Neuroinformatics and Computational Neuroscience Doctoral Training Centre, University of Edinburgh.

References

Amir, E., and Chang, A. 2008. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research* 33:349–402.

Benson, S. S. 1996. *Learning Action Models for Reactive Autonomous Agents*. Ph.D. Dissertation, Stanford University.

Croonenborghs, T.; Ramon, J.; Blockeel, H.; and Bruynooghe, M. 2007. Online learning and exploiting relational models in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 726–731.

Doğar, M. R.; Çakmak, M.; Uğur, E.; and Şahin, E. 2007. From primitive behaviors to goal directed behavior using affordances. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2007)*, 729–734.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Freund, Y., and Schapire, R. 1999. Large margin classification using the perceptron algorithm. *Machine Learning* 37:277–96.

Gil, Y. 1994. Learning by experimentation: Incremental refinement of incomplete planning domains. In *Proceedings of the 11th International Conference on Machine Learning (ICML 1994)*, 87–95.

Halbritter, F., and Geibel, P. 2007. Learning models of relational MDPs using graph kernels. In *Proceedings of the 6th Mexican International Conference on Advances in Artificial Intelligence (MICAI 2007)*, 409–419.

Khardon, R., and Servedio, R. A. 2005. Maximum margin algorithms with Boolean kernels. *Journal of Machine Learning Research* 6:1405–1429.

Khardon, R.; Roth, D.; and Servedio, R. A. 2005. Efficiency versus convergence of Boolean kernels for on-line learning algorithms. *Journal of Artificial Intelligence Research* 24:341–356.

Metta, G., and Fitzpatrick, P. 2003. Early integration of vision and manipulation. *Adaptive Behavior* 11(2):109–128.

Modayil, J., and Kuipers, B. 2008. The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems* 56(11):879–890.

Mourão, K.; Petrick, R. P. A.; and Steedman, M. 2009. Learning action effects in partially observable domains. In *Proceedings of the ICAPS 2009 Workshop on Planning and Learning*, 15–22.

Mourão, K.; Petrick, R. P. A.; and Steedman, M. 2010. Learning action effects in partially observable domains. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 973–974.

Mourão, K. 2012. *(submitted) Learning action representations using kernel perceptrons*. Ph.D. Dissertation, School of Informatics, University of Edinburgh.

Pasula, H. M.; Zettlemoyer, L. S.; and Kaelbling, L. P. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research* 29:309–352.

Rodrigues, C.; Gérard, P.; and Rouveirol, C. 2010. Incremental learning of relational action models in noisy environments. In *Proceedings of the International Conference on Inductive Logic Programming (ILP 2010)*, 206–213.

Sadohara, K. 2001. Learning of Boolean functions using support vector machines. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT 2001)*, 106–118.

Walsh, T. J., and Littman, M. L. 2008. Efficient learning of action schemas and web-service descriptions. Technical report, Dept. of Computer Science, Rutgers University.

Wang, X. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the International Conference on Machine Learning (ICML 1995)*, 549–557.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence* 171(2-3):107–143.

Zhuo, H. H.; Yang, Q.; Hu, D. H.; and Li, L. 2010. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence* 174(18):1540–1569.

Learning the distribution of the succesful grasping from simulation, a case study

Technical Report

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

Heiko Hahn
IIS, University of Innsbruck
heiko.hahn@uibk.ac.at

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

Norbert Krüger
The Maersk Mc-Kinney Moller Institute, University of Southern Denmark
norbert@mmmi.sdu.dk

Jimmy Alison Jørgensen
The Maersk Mc-Kinney Moller Institute, University of Southern Denmark
jimali@mmmi.sdu.dk

January 25, 2012

Abstract

In this technical report the effect of the sampling approaches on the accuracy of the prediction of the successful grasping is analyzed. In simulation based experiments two, a simple and a relatively complex, objects were the targets and two different type of grippers, equipped with two and three fingers, are applied. The three-finger gripper was also used in three different configurations. Sampling procedures are investigated and compared via a set of accuracy measures of a binary classification which had to learn the separation of the successful and failed grasping actions on the two objects. We also analyze the structure of the distribution by asking that how do the successful grasping actions relate to each other; do they form a connected, contiguous subset in the full set of measurements. The result of this analysis is then used to improve the prediction accuracy by applying a more adaptive sampling technique a variant of the Markov Chain Monte Carlo(MCMC) method.

1 Introduction

In this study we experimentally analyze the prediction of the success of the grasping on a relatively complex and a simple objects. The main purpose of this analysis is to find the most promising sampling and a relating learning method which can provide high quality density estimation to the probabilities of the successful grasping actions to a given object. This sampling methods have to be sufficiently accurate and the same time fast enough to be applicable in a real life scenario.

The presented learning model allows us to predict not only the success of the failure, a binary classification approach, but the probabilities of the these potential outcomes. Further details about estimating the grasping densities can be found in the following papers, [2], [8], [5], [7], [6], [3] and [4]. These papers investigate the estimation in different circumstances and from different point view, e.g. which part of the the target object is the most promising one to grasp.

The prediction methods are applied on the data sets generated by simulation on the "Dolt" and the "Box" objects, see them on Figure 1 and Figure 2. The first object is a non-convex not an easy task for grasping, the second one is a relatively simple box. To grasp the Dolt object a simple two-finger robot hand is used. The results of these experiments have been exploited to amend the sampling technique, the learning method and the simulation procedure as well. We can consider the Dolt object based experiment as a certain base line when the further research objectives are outlined. When the box object is grasped a three-finger robot hand is applied in three different configurations. The details of the data collection and of the configuration of the robot hands can be found in next Section.

In the experiments two different types of sampling are applied. The first one can be defined as a CAD model dependent method because it tries to follow the surface of the objects and produces an approximately uniform sampling relative to that. This sampling yields generally unbalanced samples where the number of successful cases are very small relative to the failed ones. With some post-processing the proportion of successful cases can be increased at the price of some distortion caused on the underlying unknown distribution of the successful cases.

The other sampling approach is a certain type of Markov Chain Monte Carlo(MCMC) sampling, which could work on an object without exploiting its CAD model, thus can be used among more general conditions. The target of this sampling is to capture the manifold of the successful grasping in the space of all, successful and failed, ones. It mainly collects failed cases when the regions of successful cases are isolated and they can not be connected without considering negative items. As a consequence this sampling technique can produce data sets more characteristic on the successful items.

Comparing the two methods we need to think about how the success

	Total number of cases	Successful cases	Sampling method	Resampled
Dataset1	311000	1149	CAD model dependent	No
Dataset1r	11423	1149	CAD model dependent	Yes
Dataset2	1367269	62269	CAD model dependent	No
Dataset2r	91116	48124	CAD model dependent	Yes
Dataset3mc	1091264	41010	MCMC	No

Table 1: Data sets generated on the Dolt object where a two-finger hand is applied in a floating environment

probability of a grasping action can depend on the shape of an object. The successful grasping requires the knowledge about a sufficiently large neighborhood of a surface point to capture the local structure. Collecting independently information on points might not able to express the inter-dependencies of those points. Therefore the sampling method should focus on the exploration of the local neighborhoods around the points where the grasping is turned to be successful.

2 Data sources

The data sets serving as input in the binary classification procedure have been generated on two, a *Dolt* and a *Box*, objects. In case of the *Dolt* object a two-finger gripper is applied in three different sampling approaches, where samples provided by two sampling methods are resampled since they contain too few positive cases, see details in Section 2.1. The third sampling is designed as a more adaptive method based on a MCMCM type method focusing on the collection of items on the manifold spanned by the successful cases. At the end we have 5 different samples see in Table 1.

To each case in the samples there are given the object relative position of the gripper in the space of $\mathbb{R}^3$, and the corresponding orientation in the space of $\mathcal{SO}(3)$ which is represented by unit quaternions. The task is to predict the potential success of the grasping by assuming the knowledge about a given pair of position and orientation.

Since the first two data sets, *Dataset1* and *Dataset2*, show up significantly different distribution, see the result below, therefore the direct comparison of the result between the data sets is difficult. It turned to be a better approach is to find an adaptive sampling procedure, e.g. a Markov Chain Monte Carlo(MCMC) type one, which can be applicable with more generality, see

	Total number of cases	Successful cases	Sampling method	Finger configuration
Dataset4	25000	9539	CAD model dependent	BALL
Dataset5	35000	9968	CAD model dependent	CYL
Dataset6	40000	10009	CAD model dependent	PAR
Dataset4mc	33101	23305	MCMC	BALL
Dataset5mc	28488	18691	MCMC	CYL
Dataset6mc	41970	12603	MCMC	PAR

Table 2: Data sets generated to the Box object by applying three-finger gripper(dexterous) in three different configuration in a floating environment.

further details about these approaches in [11] and the references therein. The size of the corresponding data set, Dataset3mc, is shown in Table 1.

In the experiments dealing with the second object, the *Box*, a three-finger gripper, *Shunk SDH dexterous robot hand*, is used in three different configuration, see their brief description in Table 3, and these configuration can be seen on the corresponding sub-images of Figure 4.

Abbreviation	Finger configuration
BALL	Ball Precision Grip
CYL	Cylinder Precision Grip
PAR	Parallel Grip

Table 3: Configuration of the three-finger Shunk SDH dexterous robot hand used in the data generation

Based on these configuration six data sets are collected, three of those are generated by sampling in a CAD model related way, and three others come from MCMC sampling. The parameters of the these data sets are presented in Table 2.

Remark 1. *In the identifiers of the data sets we apply the following rules, those identifiers ended with "mc" are from MCMC type sampling, if the suffix is "r" then it shows the data set is resampled in the way detailed in Section 2.1. If there is no alphabetic suffix after the number appearing at the end of the identifier then the data set is generated by CAD model related sampling.*

2.1 Increasing the proportion of positive cases by supervised clustering

Since the data sets, Dataset1 and Dataset2, are very unbalanced, i.e. the negative cases overwhelmingly dominate these data sets, reducing the number of negative cases with a comprehensive strategy not only decrease the computational burden but could increase the reliability of the estimation. The reduction applied preserves those negative items which occurring in the neighborhood of at least one positive item, and drops those which appear away from any positive cases.

This reduction has been carried out by a k-means clustering like algorithm. The details are the following

- Let all positive cases, (in case of Dataset1 it is equal to 1149 items), be fixed as cluster centers.
- Negative cases satisfying simultaneously the following criteria
  - at least one positive case is closer than a given distance, e.g. the *(Maximum radius of data in position)* $/(10\sqrt{3})$,
  - at least one positive case is closer in orientation than a given angle, e.g. $\pi/18$,

  are accepted, and all other negative cases are dropped.

The same procedure is applied on Dataset2 with a modification because the number of the positive items in this set is much greater. First 2000 positive items selected randomly as seeds of the procedure then the algorithm detailed above has been applied. The neighborhood of seeds might contain further positive items as well which were also included together with the adjacent negative items. At the end of this procedure we have received the data sets, Dataset1r and Dataset2r, see their parameters in Table 1.

These modifications are partially preserve the structure of the distribution around the positive cases but the whole distribution might be biased.

3 Learning method

The grasping action can be described by the gripper position $p \in \mathbb{R}^3$ and orientation $o \in \mathcal{S}(O)^3$, they serve as input data in the learning problem. The success of the grasping action is given by $y \in \{+1, -1\}$, where the 1 labels the positive, successful cases. The number of actions considered in the learning method is denoted by m. The learning method applied is a generalization of the well known Support Vector Machine to the case which can predict arbitrary, not only binary, outputs which can be represented in a properly chosen Hilbert space, e.g. the space of square integrable functions comprising the probability density functions. The learning approach

has been applied for multiclass classification in [14], for graph represented structured output learning in [1] and [12], and for learning when the data sources are incomplete [13], [10] and [9] as an alternative to the well known Expectation Maximization algorithm.

3.1 Feature representation

All sources of the information, position, orientation and success are represented in the corresponding feature spaces. These feature spaces are denoted by $\mathcal{H}_{\text{position}}$, $\mathcal{H}_{\text{orientation}}$ and $\mathcal{H}_{\text{success}}$. The functions providing the feature representations are denoted and given by

$$\begin{aligned} &\psi_s : \{+1, -1\} \to \mathcal{H}_{\text{success}} \\ &\phi_p : \mathbb{R}^3 \to \mathcal{H}_{\text{position}} \\ &\phi_o : \mathcal{S}(O)^3 \to \mathcal{H}_{\text{orientation}} \end{aligned} \tag{1}$$

The feature spaces are chosen as probability density functions of Gaussian distributions, namely

$$\begin{aligned} &\psi_s(y) = f(.|y, \sigma_{\text{success}}) \\ &\quad \Rightarrow \tfrac{1}{(2\pi)^{1/2}\sigma_{\text{success}}} e^{-\|t-y\|^2/(2\sigma^2_{\text{success}})}, y \in \{+1, -1\}, t \in \mathbb{R}, \\ &\phi_p(p) = f(.|p, \sigma_{\text{position}}) \\ &\quad \Rightarrow \tfrac{1}{(2\pi)^{3/2}\sigma^3_{\text{position}}} e^{-\|t-y\|^2/(2\sigma^2_{\text{position}})}, p, t \in \mathbb{R}^3, \\ &\phi_o(o) = f(.|o, \sigma_{\text{orientation}}) \\ &\quad \Rightarrow \tfrac{1}{(2\pi)^{4/2}\sigma^4_{\text{orientation}}} e^{-\|t-y\|^2/(2\sigma^2_{\text{orientation}})}, o \in \mathcal{S}^3 (\sim \mathcal{S}(O)^3), t \in \mathbb{R}^4, \end{aligned} \tag{2}$$

where the standard deviations σ_{success}, σ_{position} and $\sigma_{\text{orientation}}$ are independently chosen to every feature mapping in cross validation. These feature mappings lead to Gaussian kernels with parameters equal to the standard deviations of the features multiplied by $\sqrt{2}$.

In the learning method we look for a linear mapping $\mathbf{W} : \mathcal{H}_{\text{position}} \otimes \mathcal{H}_{\text{orientation}} \to \mathcal{H}_{\text{success}}$ which can approximate the functional relationship between the position, orientation and the success represented by the corresponding feature spaces. The symbol $\otimes$ denotes the tensor product of two spaces.

3.2 Optimization framework

In the optimization method we force the inner products

$$\langle \psi_s(y), \mathbf{W}(\phi_p(p) \otimes \phi_o(o)) \rangle \tag{3}$$

to be uniformly high for all training items, since if the linear mapping $\mathbf{W}$ is correct then the inner product between the prediction $\mathbf{W}(\phi_p(p) \otimes \phi_o(o))$ and the real value $\psi_s(y)$ has to take its maximum value. Based on this the primal optimization problem is formulated as follows:

$$
\begin{array}{lll}
\min & \frac{1}{2}\|\mathbf{W}\|^2_{\text{Frobenius}} + C\sum_{i=1}^m \xi_i & \\
\text{w.r.t.} & \mathbf{W} : \mathcal{H}_{\text{position}} \times \mathcal{H}_{\text{orientation}} \to \mathcal{H}_{\text{success}} & \text{linear map} \\
& \xi_i \in \mathbb{R}\ i = 1, \ldots, m & \text{item wise errors} \\
\text{s.t.} & \langle \psi_s(y_i), \mathbf{W}(\phi_p(p_i) \otimes \phi_o(o_i)) \rangle \geq 1 - \xi_i, & \\
& \xi_i \geq 0, \quad i = 1, \ldots, m, &
\end{array}
\tag{4}
$$

where $C > 0$ is a penalty constant balancing between the regularization term $\|\mathbf{W}\|^2_{\text{Frobenius}}$ constraining the class of linear mapping to achieve better regularization, and the empirical error $\sum_{i=1}^m \xi_i$.

After introducing Lagrangians $\boldsymbol{\alpha} = (\alpha_i), i = 1, \ldots, m$ to each constraints we can derive the dual problem which is finally solved instead of the primal form

$$
\begin{array}{lll}
\min & \frac{1}{2}\boldsymbol{\alpha}' & \overbrace{\mathbf{K}}^{\text{compound kernel}} \quad \boldsymbol{\alpha} - \langle \mathbf{1}, \boldsymbol{\alpha} \rangle \\
\text{w.r.t.} & \boldsymbol{\alpha} \geq \mathbf{0} & \\
\text{s.t.} & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, &
\end{array}
\tag{5}
$$

where $\mathbf{1}$ is vector with components equal to 1, and the compound kernel $\mathbf{K}$ is the point wise product of the kernels belonging to the three feature spaces:

$$
K_{ij} = (K_{\text{success}})_{ij}(K_{\text{position}})_{ij}(K_{\text{orientation}})_{ij}, i, j = 1, \ldots, m, \tag{6}
$$

where

$$
\begin{array}{l}
(K_{\text{success}})_{ij} = \langle \psi_s(y_i), \psi_s(y_j) \rangle, \\
(K_{\text{position}})_{ij} = \langle \phi_p(p_i), \phi_p(p_j) \rangle, \\
(K_{\text{orientation}})_{ij} = \langle \phi_o(o_i), \phi_o(o_j) \rangle.
\end{array}
\tag{7}
$$

After solving the dual problem for $\boldsymbol{\alpha}$ we have

$$
\mathbf{W} = \textstyle\sum_{i=1}^m \alpha_i(\psi_s(y_i) \otimes \phi_p(p_i) \otimes \phi_o(o_i)) \tag{8}
$$

A prediction in the feature space of success can be computed for a given pair of position p and orientation o by applying

$$
\begin{array}{l}
\psi_s(y) = \mathbf{W}(\phi_p(p) \otimes \phi_o(o)) \\
= \sum_{i=1}^m \alpha_i \psi_s(y_i) \langle \phi_p(p_i), \phi_p(p) \rangle \langle \phi_o(o_i), \phi_o(o) \rangle,
\end{array}
\tag{9}
$$

which gives a non-negative linear combination of density functions corresponding to the features of success in the training set. This can be interpreted as an unnormalized density mixture, hence after normalization it can be used to estimate a confidence interval around the predicted value.

To derive the success label $\{+1, -1\}$ we might apply

$$
\begin{array}{l}
\hat{y} = \arg\max_{y \in \{+1,-1\}} \langle \psi_s(y) \mathbf{W}(\phi_p(p) \otimes \phi_o(o)) \rangle \\
= \sum_{i=1}^m \alpha_i \langle \psi_s(y), \psi_s(y_i) \rangle \langle \phi_p(p_i), \phi_p(p) \rangle \langle \phi_o(o_i), \phi_o(o) \rangle.
\end{array}
\tag{10}
$$

TP	True positive
FP	False positive
FN	False negative
TN	True negative
Precision	$TP/(TP+FP)$
Recall	$TP/(TP+FN)$
F1	$(2*precision*recall)/(precision+recall)$
Accuracy	$(TP+TN)/(TP+FP+FN+TN)$
EER	Equal error rate in receiver operating characteristic (ROC)
AUC	Area under curve of receiver operating characteristic (ROC)
5-NN	stands for 5-Nearest Neighbors method

Table 4: Measures and their components used in the prediction method

4 Experimental Results

The measures used in the prediction of the successful grasping are described in Table 4.

We apply not only one measure, the accuracy, since the data sets are very unbalanced, might not capture correctly the capability of the prediction of the positive cases, for example predicting in such a way that all test items belong to the negative cases can give misleadingly high accuracy. The measures: *Area under curve(AUC)*, *Precision*, *Recall* and their combination *F1*, see the definitions in Table 4, can reflect better how well the predictor can behave on small number of positive cases. In this study we focus on the AUC measure, namely the kernel parameters are optimized by maximizing the AUC in cross validation.

The results are computed on a uniform random subsample of the original sets via 5-fold cross validation. The kernel parameters are optimized by further splitting the training set into validation training and validation test sets. The results are summarized in the following tables.

4.1 Dolt object, two-finger hand

The tables presented here display the prediction results computed for the Dolt object based on different sampling methods. We firstly computed the estimations on the subsampled data sets Dataset1r and Dataset2r.

	Precision	Recall	F1	Accuracy	EER	AUC
Dataset1r	0.3113	0.6008	0.4088	0.8258	0.6842	0.5110,

Average predicted frequencies, normalized to 100

Dataset1r		Observed	
		Positive	Negative
Predicted	Positive	TP = 6.04	FP = 13.39
	Negative	FN = 4.03	TN = 76.54,

	Precision	Recall	F1	Accuracy	EER	AUC
Dataset2r	0.6791	0.6783	0.6762	0.6567	0.6428	0.4257,

Average predicted frequencies,normalized to 100

Dataset2r		Observed	
		Positive	Negative
Predicted	Positive	TP = 36.14	FP = 17.14
	Negative	FN = 17.14	TN = 29.57.

Dataset2r is a more balanced one which gives significantly higher values in the precision and the recall measures. The higher accuracy in Dataset1r is mostly the consequence of the very high proportion of negative items, they occur approximately 10 times more than the positives cases.

The first one shows the distribution in the $\mathbb{R}^3$ and the second one in 3D projection of the $\mathcal{S}(O)^3$ space. The points belongs to four classes of true positives, false positives, false negatives and true negatives.

In case of the third data set, Dataset3mc we have the following result,

	Precision	Recall	F1	Accuracy	EER	AUC
Dataset2	0.0841	0.8739	0.1533	0.6654	0.7176	0.5741
Dataset3mc	0.1165	0.8984	0.2060	0.7239	0.7556	0.6439,

here we omitted the subsampling which increases the proportion of the positive items. To make the results comparable we also display the corresponding accuracy measures for the original Dataset2, whose size is approximately the same as Dataset3mc, computed at the same conditions.

Average predicted frequencies, normalized to 100

Dataset3mc		Observed	
		Positive	Negative
Predicted	Positive	TP = 26.32	FP = 20.01
	Negative	FN = 2.99	TN = 50.68.

Since the positive cases constitute approximately 4% of the data set, the precision measure is low but if we compare the result to Dataset2 the improvement is clear. When those measures, EER and AUC, are considered which suffer much less from the unbalanced classes we receive significantly better results. Hence applying a sampling procedure which does not require

a CAD model is a significantly more promising approach for a robot system which has to explore an environment consisting of new, earlier unseen, objects.

4.2 Box object, three-finger hand

From the Tables presented here we can conclude that the three-finger hand produces significantly high accuracy on the simple box object in all applied measures. Furthermore we can recognize that the MCMC sampling performs much better than that is adapted to the CAD model.

4.2.1 BALL finger configuration

	Precision	Recall	F1	Accuracy	EER	AUC
Dataset4	0.7846	0.7781	0.7812	0.8314	0.7950	0.6731
Dataset4mc	0.9115	0.9283	0.9197	0.8859	0.8170	0.7288

Average predicted frequencies,normalized to 100

Dataset4		Observed	
		Positive	Negative
Predicted	Positive	TP = 30.09	FP = 8.27
	Negative	FN = 8.59	TN = 53.05

Average predicted frequencies,normalized to 100

Dataset4mc		Observed	
		Positive	Negative
Predicted	Positive	TP = 65.34	FP = 6.36
	Negative	FN = 5.06	TN = 23.25

4.2.2 CYL finger configuration

	Precision	Recall	F1	Accuracy	EER	AUC
Dataset5	0.7947	0.7109	0.7503	0.8633	0.7631	0.6579
Dataset5mc	0.9576	0.9257	0.9413	0.9122	0.8745	0.8041

Average predicted frequencies,normalized to 100

Dataset5		Observed	
		Positive	Negative
Predicted	Positive	TP = 20.52	FP = 5.31
	Negative	FN = 8.35	TN = 65.82

Average predicted frequencies,normalized to 100

Dataset5mc		Observed	
		Positive	Negative
Predicted	Positive	TP = 70.48	FP = 3.13
	Negative	FN = 5.66	TN = 20.73

4.2.3 PAR finger configuration

	Precision	Recall	F1	Accuracy	EER	AUC
Dataset6	0.7847	0.6822	0.7298	0.8736	0.7441	0.6395
Dataset6mc	0.8509	0.8393	0.8450	0.8423	0.8383	0.7095

Average predicted frequencies,normalized to 100

Dataset6		Observed	
		Positive	Negative
Predicted	Positive	TP = 17.08	FP = 4.68
	Negative	FN = 7.96	TN = 70.28

Average predicted frequencies,normalized to 100

Dataset6mc		Observed	
		Positive	Negative
Predicted	Positive	TP = 43.00	FP = 7.54
	Negative	FN = 8.23	TN = 41.23

5 Investigation of the distribution of the positive cases in CAD model dependent sampling

The small number of positive cases (=1149) in relation to the entire data set (=311000) in Dataset1 rises the question how the positive cases are distributed among the vast majority of negative ones. To this end the nearest neighborhood(NN) of each of the positive cases are analyzed. The size of these neighborhoods was set to 5 in the computation of the results presented in the following tables.

The next tables display the local distribution of the positive and negative items in the neighborhoods. First the number of positive items are counted, then the number of those positive cases are enumerated whose neighborhood contains no any other, or 1,2,3,4,5 positive cases. The enumeration was carried out for the positions and the orientations separately.

Dataset1, 1149 positive cases

Number of positive cases in 5-NN	0	1	2	3	4	5
Positive cases with as many positive neighbors as shown in the first row in $\mathbb{R}^3$	899	212	37	1	0	0
In percentage	78.24	18.45	3.22	0.09	0.00	0.00
Positive cases with as many positive neighbors as shown in the first row in $SO(3)$	978	156	14	1	0	0
In percentage	85.12	13.58	1.22	0.09	0.00	0.00

This result shows that almost all positive cases are singular in Dataset1, i.e. they lie isolated among the negative ones. It might imply that there is no contiguous range of the position and orientation space where the positivity is a characteristic property in that sample..

Dataset2, 62269 positive cases

Number of positive cases in 5-NN	0	1	2	3	4	5
Positive cases with as many positive neighbors as shown in the first row in $\mathbb{R}^3$	25222	18978	11568	4971	1368	162
In percentage	40.50	30.48	18.58	7.98	2.20	0.26
Positive cases with as many positive neighbors as shown in the first row in $SO(3)$	26678	18415	11253	4654	1142	127
In percentage	42.84	29.57	18.07	7.47	1.83	0.20

In Dataset2 the nearest neighbors of the positive items contain significantly more positive items, but if we consider the majority rule, where the number of positive neighbors greater or equal to 3, then most of the positive cases fail to be predicted as positive ones based on a 5-NN predictor.

5.1 Conclusion

Based on this analysis we decided to change the strategy of the sampling. The proposed new approach, an MCMC type method, concentrates on the manifold of the successful items and try to avoid segments of the sampling space containing no success items with high probability.

Acknowledgment

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.

A Sampling by a variant of the Metropolis-Hasting method

To implement an adaptive sampling method which highly independent from the shape of the target object, a Markov Chain Monte Carlo type method, a variant of the Metropolis-Hastings algorithm is applied. This algorithm is fundamentally a proposal distribution based random search algorithm in

the space of the observed events. Our objective for applying this kind of algorithm is to capture the manifold of the successful cases in the space of all, successful and failed grasping actions.

A.1 Description of the algorithm

Let the set $\mathcal{G} = \{g_1, \cdots, g_{n_p}\}$ be the collection of all successful grasping. These grasping are given by their position and orientation as usual.

1. Input:
   - $\mathcal{G}$ set of known positive grasping,
   - radius R in $\mathbb{R}^3$, for example $R = 1cm$,
   - angle α defining a cone in $SO(3)$, for example $\alpha = 15°$,
   - fix $sample\_size$, and let $isample = 0$.
2. Let $\mathcal{S} = \mathcal{G}$ be the set of successful grasping at the start point of the algorithm.

   Let $\mathcal{F} = \emptyset$ be the set of failures which is empty at the start point.
3. Choose uniformly one positive item s of the set $\mathcal{S}$ of successful grasping, and let $s_{position}$ be the corresponding position and $s_{orientation}$ be the corresponding orientation.
4. Take a sample item
   - $x_{position}$ from a 3 dimensional Gaussian distribution $\mathcal{N}(s_{position}, \Theta)$, where Θ is a covariance matrix with only diagonal elements equal to R^2.

     Let $P(x_{position})$ be the probability of the sample item with respect to the Gaussian distribution $\mathcal{N}(s_{position}, R^2)$.
   - and $x_{orientation}$ from 4 dimensional Von Mises-Fisher distribution $C_4(\kappa)\exp(\kappa \langle s_{orientation}, x\rangle)$, where $\kappa = \alpha$.

     Let $P(x_{orientation})$ be the probability of the sample item with respect to Von Mises-Fisher distribution $C_4(\kappa)\exp(\kappa \langle s_{orientation}, x\rangle)$.
5. - If $x_{position} <= R$, and the angle between $x_{orientation}$ and $s_{orientation}$ is $\leq \alpha$ then
     - if the grasping is successful at $(x_{position}, x_{orientation})$ then add this item to $\mathcal{S}$,
     - if the grasping is unsuccessful at $(x_{position}, x_{orientation})$ then add this item to $\mathcal{F}$,
   - Otherwise

- If the grasping is successful at $(x_{position}, x_{orientation})$ then add this item to $\mathcal{S}$,
- Otherwise
  Accept the point $(x_{position}, x_{orientation})$ with probability $P(x_{position})P(x_{orientation})$, and add this item to $\mathcal{F}$,

6. If the item is not accepted then go to **Step 4** and repeat the sampling of x around the same positive item s !

7. $isample+ = 1$

   If $isample < sample\_size$ Go to **Step 3**,

   otherwise Stop and return $\mathcal{S}$ and $\mathcal{F}$!

References

[1] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, and J. Rousu. Towards structured output prediction of enzyme function. In *BMC Proceedings, 2(Suppl 4):S2.* 2008.

[2] Emre Başeski, Nicolas Pugeault, Sinan Kalkan, Justus Piater, and Norbert Krüger. Using Multi-Modal 3D Contours and Their Relations for Object Encoding and Grasping. In *24th International Symposium on Computer and Information Sciences*, 9 2009. Northern Cyprus.

[3] Leon Bodenhagen, Renaud Detry, Justus Piater, and Norbert Krüger. What a successful grasp tells about the success chances of grasps in its vicinity. In *ICDL-EpiRob.* IEEE, 2011.

[4] Renaud Detry, Carl Ek, Marianne Madry, Justus Piater, and Danica Kragić. Generalizing Grasps Across Partly Similar Objects. In *International Conference on Robotics and Automation*, 2012. To appear.

[5] Renaud Detry, Dirk Kraft, Oliver Kroemer, Leon Bodenhagen, Jan Peters, Norbert Krüger, and Justus Piater. Learning Grasp Affordance Densities. *Paladyn Journal of Behavioral Robotics*, 2(1):1–17, 2011.

[6] Renaud Detry and Justus Piater. Grasp Generalization Via Predictive Parts. In *Austrian Robotics Workshop*, 5 2011.

[7] Benedikt Hupfauf, Heiko Hahn, Leon Bodenhagen, Dirk Kraft, Norbert Krüger, and Justus Piater. Grasp Densities for Grasp Refinement in Industrial Bin Picking. In *Workshop on Uncertainty in Automation*, 5 2011. Workshop at ICRA.

[8] Dirk Kraft, Renaud Detry, Nicolas Pugeault, Emre Başeski, Frank Guerin, Justus Piater, and Norbert Krüger. Development of Object and Grasping Knowledge by Robot Exploration. *IEEE Transactions on Autonomous Mental Development*, 2(4):368–383, 12 2010.

[9] Ghazanfar M.A., Prugel-Bennett A., and Szedmak S. Kernel mapping recommender system algorithms. *Information Sciences Journal*, 2011. Accepted.

[10] Ghazanfar M.A., Szedmak S., and Prugel-Bennett A. 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.

[11] D.J.C. Mackay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, UK, 2003.

[12] S. Szedmak and Z. Hussain. A universal machine learning optimization framework for arbitrary outputs. 2009. http://eprints.pascal-network.org.

[13] S. Szedmak, Y. Ni, and S. R. Gunn. Maximum margin learning with incomplete data: Learning networks instead of tabels. *Journal of Machine Learning Research, Proceedings*, 11, Workshop on Applications of Pattern Analysis:96–102, 2010. jmlr.csail.mit.edu/proceedings/papers/v11/szedmak10a/szedmak10a.pdf.

[14] S. Szedmak, J. Shawe-Taylor, and E. Parado-Hernandez. Learning via linear operators: Maximum margin regression. In *PASCAL Research Reports, http://eprints.pascal-network.org/*. 2005.

B Figures

Figure 1: The original Dolt object whose CAD model is used in the simulation

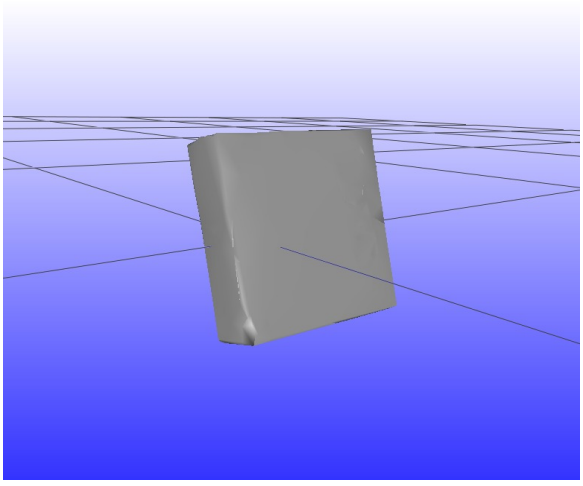


Figure 2: The CAD model of the box object

Dolt

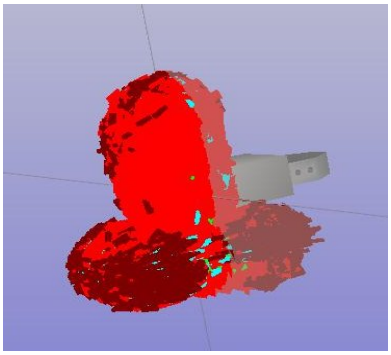


Figure 3: Distribution of the grasping success in position space, $\mathbb{R}^3$, based on Dataset1 generated by two-finger gripper, green = success, blue= slipped, light red = dropped, dark red = missed

Box by BALL config. Box by CYL config. Box by PAR config.

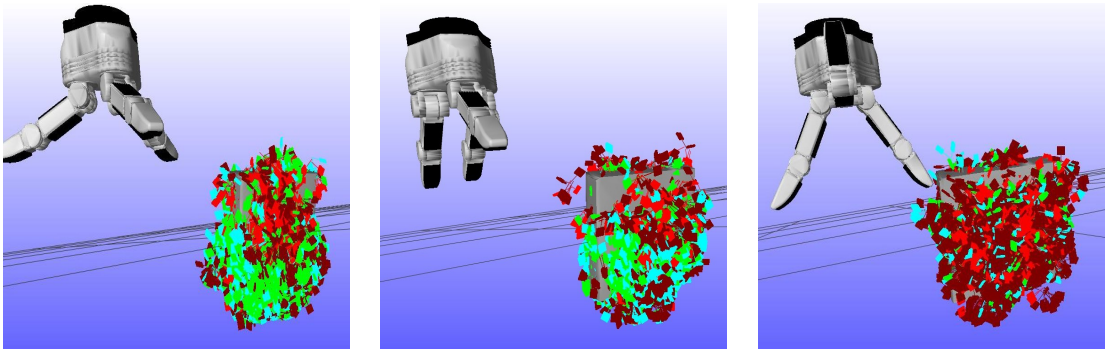


Figure 4: Distribution of the grasping success in position space, $\mathbb{R}^3$, based on data sets: Dataset4mc, Dataset5mc and Dataset4mc generated by three configurations of the three-finger gripper, green = success, blue= slipped, light red = dropped, dark red = missed

Robotics Group
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark

Technical Report no. 2012 – 1

Introduction to feature to grasp association

Mikkel Tang Thomsen [mitho07@student.sdu.dk]

February 2, 2012

Title	Introduction to feature to grasp association
Author(s)	Mikkel Tang Thomsen [mitho07@student.sdu.dk]
Publication History	

Abstract

This journal serves as an introduction to the feature to grasp space. The main objective is to investigate the feature to grasp association space in terms of the ECV (Early Cognitive Vision) high level features of 3D contours and 3D surfaces for single features. This involves simulating grasps with a two finger parallel gripper in a number of scenes with features of interest. A variety of situations have been evaluated exposing the feature to grasp space. All the situations are held up against the ground truth feature reference frame and a ECV feature extracted reference frame showing that the surface extraction in particular seems promising whereas the contour features introduces certain issues. A qualitative analysis has been performed on the acquired results.

1 Introduction

This journal covers the work in connection with an investigation of the feature to grasp association. In the following sections the project will be motivated introduced and related to previous work.

1.1 Motivation

Grasping of unknown objects has an increasing interest in the robotic community. One of the primary reasons for this is the increasing integration of robots in our human world, e.g. robotic vacuum cleaners. In order to take the next steps in autonomous robotics it is necessary for robots to be able to interact with our world. One of the primary interactions for humans is the ability to grasp things. This however is very complicated due to the complicated world we interact with. Furthermore humans have a grasping mechanism which is highly sophisticated through thousand of years of evolution.

By introducing a visual system and base grasps on visual cues, it has previously been shown that a high probability of a successful grasp can be achieved, when utilising simple relation between two feature and corresponding simple predefined grasps. From this results it is proposed that other and higher order relations exist in between multiple features, that yield higher probability of successful grasps. In a search for such higher order relations the grasp space for a given feature set must be spanned and a search in the cross space of visual features and grasps must be carried out. In this project the initial steps towards the objective are taken by investigating the grasp space in terms of single specific visual features and thereby acquiring an initial understanding of the feature-grasp cross space.

This projects serves as an introduction to a Master Thesis with the purpose of utilizing the grasp space for a given object and corresponding ECV (Early Cognitive Vision) features to search for higher order relation in between features that increases the probability of a successful grasp.

1.2 Overview

The problem statement can be described in the following sentences.

1. Investigate the feature to grasp association by means of 3D contours and 3D surfaces and associated simulated grasps utilizing the grasp simulator in RobWork.

2. Show qualitatively that a correlation exists in between feature space and grasp space by investigating the spanned feature-grasp space

The project consists of a number of different aspects each contributing to the overall objective of investigating the feature to grasp association. In figure 1 an overview of the different aspects can be seen. The key elements are divided into a RobWork domain part, consisting of a visualization and simulation environment, enabling stereo camera image acquisition and grasp simulation, and a ECV domain enabling feature extraction based on the given stereo images and camera parameters. As an input a number of scenes or situation with specific features of interest are designed and used. Finally a qualitatively analysis is carried out based on the acquired simulation and extraction data.

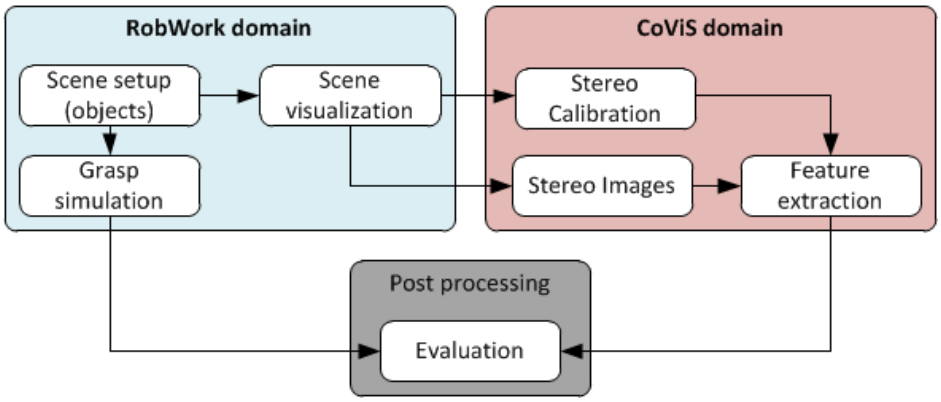
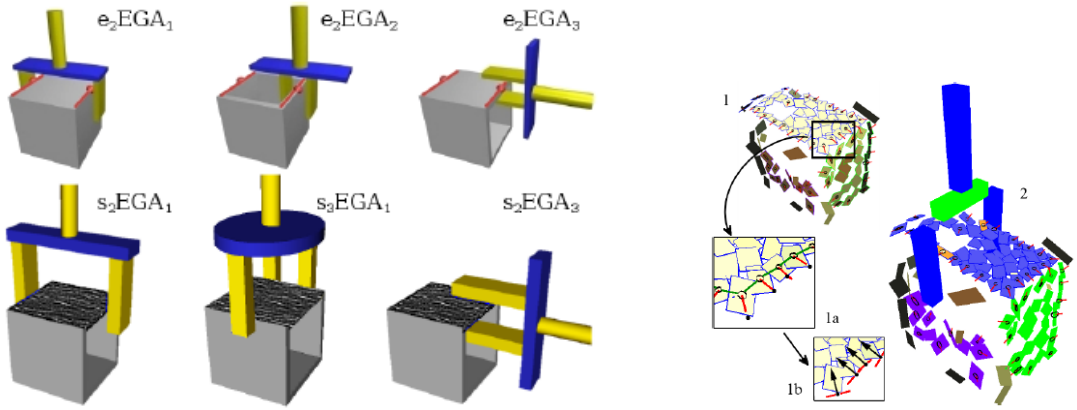


Figure 1: Overview of the different aspects in the project.

1.3 Related Work

The topic relates to the work conducted by Mila Popović in [5] and [4], where it was proposed that a number of predefined simple grasp types triggered by ECV features would yield a good probability of a successful grasp. In figure 2 the different grasp types are shown respectively for contour grasps and surface grasp. Through a number of experiments with real world data it was shown that such predefined simple grasp gave a high probability for being a successful grasp. In this context the results are interesting as they should be visible in the feature to grasp space that is spanned, and by qualitative analysis justified.



(a) (b)

Figure 2: 2(a) introducing the proposed simple grasp. 2(b) showing a visualization of three extracted surfaces and one of the corresponding grasps. Both illustrations have been taken from [4].

2 Theory

In this section, the theoretical foundation for the feature to grasp association will be established. This involves a brief introduction to ECV framework, RobWork simulation environment and the grasp representation.

2.1 Early Cognitive Vision feature framework

The visual foundation of the feature grasp association is the ECV (Early Cognitive Vision) framework feature set, see [1] for an overview of the framework. The ECV feature framework is a hierarchical representation of visual information. The framework is relying in stereo images, from which two different hierarchies evolves, namely a contour and surface part. Each of these parts is layered in a hierarchy where the level of abstraction is increased. The level of abstraction starts out with 2D feature and ends at the highest level of abstraction as 3D contours and 3D surfaces. In this context the focus is on the high level features, because we want to utilize the level of abstraction that is in the ECV framework to simplify the relation between a successful grasp and multiple features.

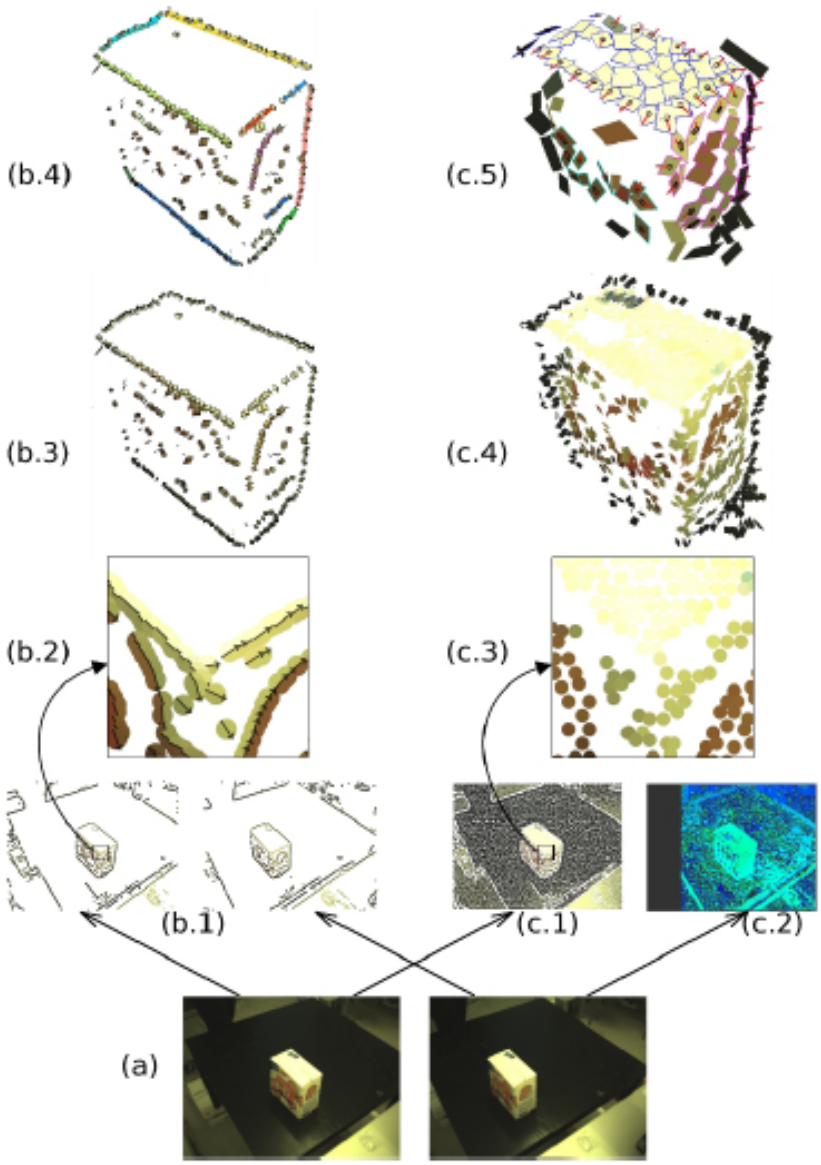


Figure 3: Illustration of the layered hierarchical structure of the ECV feature set. Evolving from stereo images to high level features of 3D contours and 3D surfaces. Taken from [4].

The 3D contour and 3D surface are formally introduced in equation 1 and 2, where P depicts the position, Θ the orientation, C the colour distribution and D the dimension of the feature.

$$\Pi_{contour} = (P, \Theta, C, D) \tag{1}$$

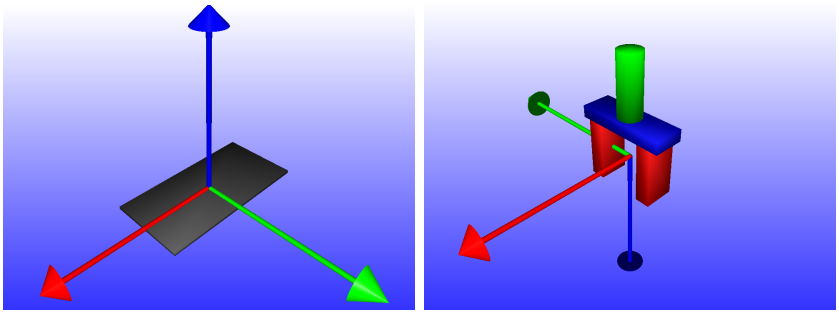
$$\Pi_{surface} = (P, \Theta, C, D) \tag{2}$$

The extraction process of the position, orientation and dimension are based on the hierarchical structure, by utilising the sub-features from the layer below. Given a set of sub-features the position is derived by calculating the centre of mass of these sub-features. In a similar way the sub-features are used to calculate the three main direction of this sub-feature cloud using principal components analysis. The largest component is the x-axis the second largest the y-axis and the smallest the z-axis giving a right hand coordinate system describing the orientation. For a 3D surface this seems reasonable, namely that the two largest components spans the surface and the smallest depicts the normal to the surface. For the 3D contour it is more ambiguous due to the fact that a straight contour ideally is a line hence the orientation is undefined except from the x-axis which will be along the contour. In the rest of this journal contour and 3D contour will be used interchangeably for a 3D contour whereas surface and 3D surface will be used interchangeably for 3D surfaces.

2.2 Grasp representation

A grasp is given as a 6D homogeneous transformation from a reference frame to a gripper frame and an associated grasp value which depicts the success of the grasp. In simulation this could be discrete values, successful or non-successful, but it could also take other forms such as a quality measure or a probability of the success of the grasp. In equation 3 a formal notation is introduced and in figure 4 the principle is shown.

$$Grasp = \{T_{Feature}^{Gripper}, success\} \tag{3}$$



(a) (b)

Figure 4: Example of a grasp being the homogeneous transformation from the feature frame, 20(a), to the gripper frame, 20(b). A frame is described with three axes, x (red), y (green), z (blue).

2.3 Simulation in RobWork

For the grasp simulations that are to be performed as a part of the investigation of the feature to grasp space the robotic framework of RobWork [2] is used. For simulation purposes the RobWorkSim module [3] is used which is an embedded part of the RobWork framework. RobWorkSim is tailored for grasp simulations utilizing an underlying physics engine to simulate the forces and torques that are applied when performing a grasp.

3 Experiment

In this section the performed experiments will be described in terms of experimental procedure and setup.

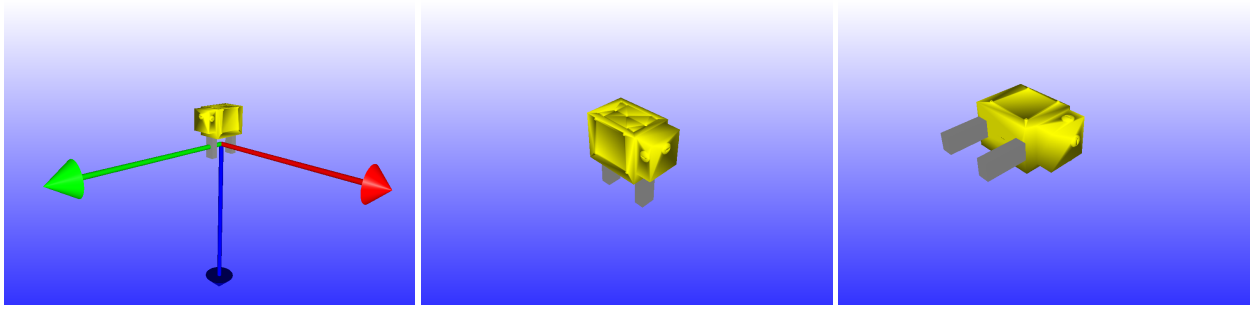
3.1 Scene setup, objects and gripper

When a grasp simulation is performed, a number of decisions have to be taken which affect the result of the simulation. The overall scene setup, the gripper type, the objects and how the grasps are sampled in the scene.

In this particular experiment the focus is in the grasp distribution around an feature. Therefore a free-floating environment is found suitable. A free floating environment meaning an environment with only the gripper and the object of interest and no outer forces affecting the object or the gripper such as gravity.

3.1.1 Gripper - Schunk PG70

A rather simple two-finger parallel gripper, Schunk PG70 [6], has been chosen for the grasp simulation. The reason for this is that the complexity of the experiment should be sufficiently low such that the results can be interpreted easily. In figure 5 the gripper is shown. The distance between the gripper jaws are set to be in the range from 0.00 m to 0.07 m.



(a) (b) (c)

Figure 5: PG70 gripper used for the experiments, 5(a) shows the gripper TCP-frame(Tool Center Point), whereas 5(b) and 5(c) shows the gripper from different angles.

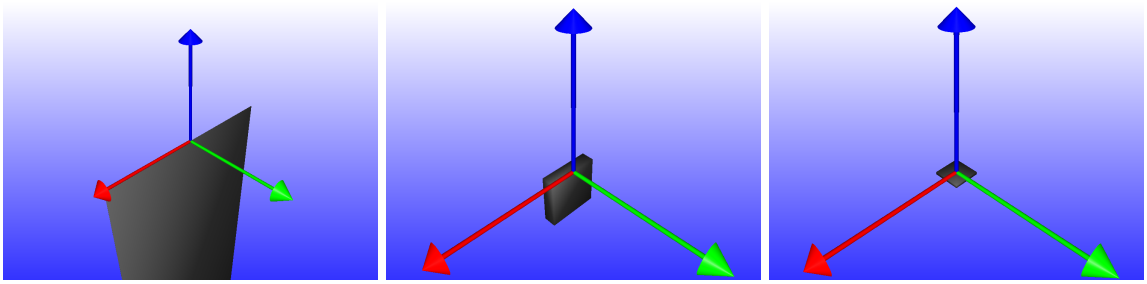
3.1.2 Objects

When choosing suitable objects for the experiments, the ECV feature framework plays a central role as it is the foundation for the feature to grasp association. To utilize the ECV feature hierarchy the top level features are the starting point for the experiments, namely surface features and contour features. To evaluate the features they are set into specific context which make sense. Based on common sense a set of situation are designed to cover some of the different situation where features will occur, see figure 6 for a visualization.

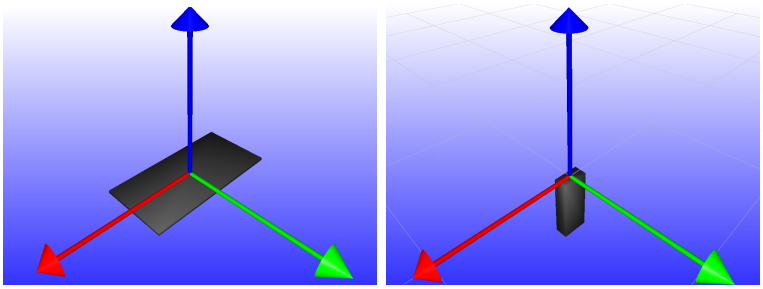
In the following subsections the different situation are presented. The dimensions of the different objects consisting the feature, are presented in table 1. The reason why the dimensions are interesting, is that the resulting grasps are closely connected to the dimensions of the objects and the properties of the gripper. By choosing different object sizes this correlation should be exposed.

Dimensions	x [m]	y [m]	z [m]
Contour on surface edge	0.1	0.001	0.25
Contour on box	0.2	0.05	0.2
Small surface	0.05	0.5	0.01
Large surface	0.5	0.25	0.01
Surface on box	0.05	0.1	0.2

Table 1: Dimension of the different test objects, denoted with the feature situation that they are related to.



(a) (b) (c)



(d) (e)

Figure 6: Visualization in RobWork of the five different situations with the embedded ground truth reference frame. Contour on edge situation in 6(a), Contour on box situation 6(b), small surface situation in 6(c), large surface situation in 6(d) and the surface on box situtation in 6(e).

Contour features

To evaluate the grasps of contours two different situations have been designed. The first situation is shown in figure 6(a), where a contour should be on the edge of a surface. This resembles a typical situation in a real application where the surface and the contour could be a part of an open box.

The second situation is a contour on the outer corner of a box, as can be seen in figure 6(b). The object, which the contour is a part of is dimensioned such that the gripper actually can grasp the object in a certain direction, which should be exposed in the experimental results later on.

Surface features

In order to investigate the grasp association with surface features three different examples with surfaces have been designed. Each of the example object is chosen because of their different dimension and relation to other surfaces.

The basic surface to grasp association are investigated trough a single surface feature, either a small or large surface, as seen in figures 6(c) and 6(d).

The difference between the surfaces are their dimensions and hence the results should be considerable different due to the gripper properties.

The third situation is one where the surface is the top of a box, as seen in figure 6(e). This objects resembles perhaps the most common situation of a surface, where it is paired with surfaces around it. The dimensions of the surface feature are set such that the gripper has the physical possibility of making a successful grasp in one direction and not in the others.

3.2 Experimental procedure

The experiments will be performed according to the following procedure.

1. Create dynamic RobWork scene with object of interest.
2. Generate 100,000 uniformly sampled grasp attempts around the feature of interest. 100.000 grasp attempts are used as it results in at least 1.000 successful grasps for every situation.
3. Simulate 100,000 grasp attempts.
4. Extract successful grasp poses.
5. Evaluate feature to grasp association.

3.2.1 Sampling strategy

In order to generate 100,000 random grasp attempts a sampling strategy is adopted. The sampling strategy is described in the following steps.

1. A position is chosen randomly in the vicinity of the 3D CAD model surface.
2. Orientation is sampled such that the direction is towards the 3D CAD model.
3. The positional part is constrained such that the focus is in the ECV feature of interest.

When introducing constrains to the sampling a number of issues are solved whereas other occur. The main issue with the sampling is that the standard sampler in RobWork is developed for sampling a full 3D object, enabling a full grasp representation for an entire object. In this context however one only wants sampling near the feature of interest. To achieve this a filter is introduced, which filters out any grasp, which position is 0.001 m in the negative z-direction of the feature reference frame. The problem with this filter is that potential good grasp can be discarded with a too strict filter, whereas grasps that do not relate to the feature of interest can be found if a too soft filter is used. Based on this assumption it was considered that a rather strict filter were to be used, as it was considered important only to have grasps that relates to the feature of interest.

3.2.2 Feature reference frame

Two options have been considered, when evaluating the grasp feature association in between a ECV high level feature and a grasp.

1. Base the feature to grasp association in a ECV extracted feature pose.
2. Base the feature to grasp association in the ground truth pose.

The first option is dependant on the feature extraction and hence dependant on the stereo images acquired from the simulated scene. This means that the position and orientation of the feature outcome is sensitive to the scene setup, hence the generalization can be lost in a biased or difficult scene for the feature extraction.

The argument that talks in favour of the feature extraction is the fact that in the real world the ground truth is not accessible as it is in simulation and hence the real data will be, at least more, similar to those acquired through the feature extraction.

The second option has the advantage that by having a coordinate system fixed to the feature ground truth principal axes the interpretation of the outcome is to some degree simplified. Furthermore the second option can be said to emulate the proposed functionality of the ECV feature space and thereby provide a generalized picture of the feature to grasp space.

4 Results

In this section the experimental results are presented one by one for the different situations presented in section 3. Furthermore the grasp simulation statistics will be presented and finally ECV extracted features to grasp association will be compared to the ground truth results.

4.1 Grasp Simulation statistics

Table 2 depicts the grasp statistics for the different situations which have been evaluated. The statistics shows how different the situations are in terms of grasp-ability as the success-rate of the grasp actions spans from 1.4 % to 20.8 % for respectively the large surface and the small surface. One thing that is not present in the statistics is the simulation time, which is highly dependant in the amount of successful grasps. This fact is reflected in the lower amount of tried grasps for the small surface as the time consumption was increasing compared to the less successful situation, due to the high success-rate. The collisions show how often the gripper is found in an initial collision

with the object, failed denotes when the gripper fails to grasp the object, slipped means that the object has slipped and simulation failure denoting when the physics engine failed to calculate the physics. As with the successes the other figures are dependant on the shape and size of the object. In this context the slipped grasp has been discarded in the results although they could be regarded as successful grasps. The success-rate is defined as follows:

$$\text{Succ.-rate} = \frac{\text{Succ.}}{\text{No. grasp attempts}}\% \tag{4}$$

	No. grasp attempts	Succ.	Succ.-rate	Collision	Failed	Slipped	Sim failure
ContourOnEdge	100,000	7,365	7.4 %	57,504	28,594	88	6,443
ContourOnBox	100,000	1,691	1.7 %	71,691	17,312	395	8,907
Small surface	50,000	10,396	20.8 %	25,199	9,656	1,625	3,123
Large surface	100,000	1,345	1.3 %	88,677	5,456	135	4,390
Surface on box	100,000	1,807	1.8 %	68,937	18,208	411	10,636

Table 2: Grasp simulation statistics for the different situations.

4.2 Feature to grasp association

The simulated grasp results are visualized in two different ways. A intuitively visualization is shown in RobWork, where 500 samples of the successful grasp are visualised by inserting the gripper shown in figure 7 in a scene with the evaluated situation, hence giving an initial understanding of the distribution of grasp around the feature of interest. An example of this is seen in figure 10, where different views are shown.

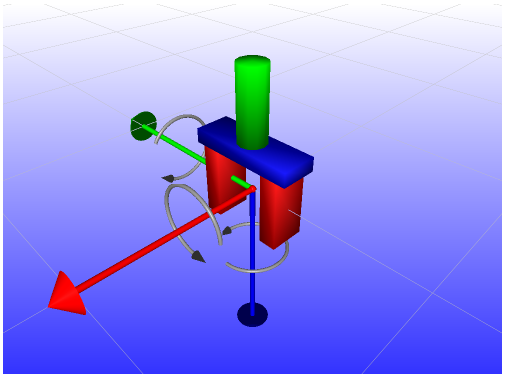
As a second visualization, projections of the 6D grasp space for the successful grasps are shown in a number of histograms. The histograms are based on the 6D homogeneous transformation matrix of a successful grasp, see equation 5. In the matrix r's denote the rotational parts whereas p's denote the positional part.

$$T_{\text{grasp}} = \begin{bmatrix} r_{x,x} & r_{x,y} & r_{x,z} & p_x \\ r_{y,x} & r_{y,y} & r_{y,z} & p_y \\ r_{z,x} & r_{z,y} & r_{z,z} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

By plotting the different components of the transformations matrix, projections of the 6D space can be visualized. The projections are shown in histograms which depicts the distribution in percent of the successful grasps in the 6D space, see figure 11 for an example.

Intuitively the histograms show the projection of the three main axis of the gripper projected into the feature reference frame. Meaning that the vector $x = [r_{x,x}, r_{y,x}, r_{z,x}]^T$ depicts the orientation of the the gripper x-axis in terms of the feature reference frame, $y = [r_{x,y}, r_{y,y}, r_{z,y}]^T$ the y-axis and $z = [r_{x,z}, r_{y,z}, r_{z,z}]^T$ the z-axis.

In figure 7 the three main rotational axis of the gripper are illustrated. The illustration will serve as reference point when explaining the feature to grasp association results.



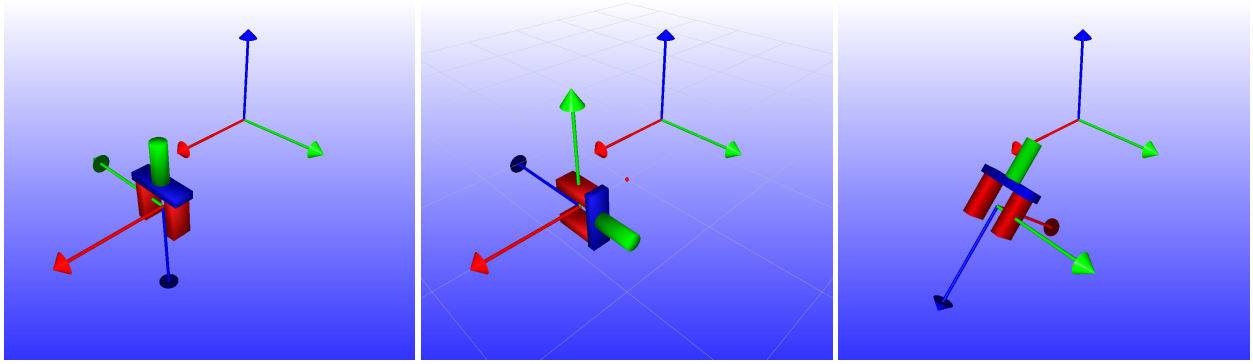
(a)

Figure 7: Rotation around the three main axes of the gripper.

4.2.1 Rotational histogram interpretation

As an additional introduction to the histograms, three examples of the rotational part of a transformation matrix is presented and visualized with a gripper in RobWork. In equation 6 three different rotations matrices are presented. In figure 8 the corresponding gripper orientation with respect to a reference frame are visualised. In a corresponding histogram the three rotations are shown, see figure 9, to get a feeling of the mapping.

$$R_{\mathrm{g},1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} R_{\mathrm{g},2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} R_{\mathrm{g},3} = \begin{bmatrix} -0.7071 & 0 & 0.7071 \\ 0 & 1 & 0 \\ -0.7071 & 0 & -0.7071 \end{bmatrix} \tag{6}$$



(a) (b) (c)

Figure 8: Visualisation of three different orientations for a gripper with respect to a reference frame, given by a rotation matrix. The reference frame is in the background and the gripper frame is visualised in the gripper.

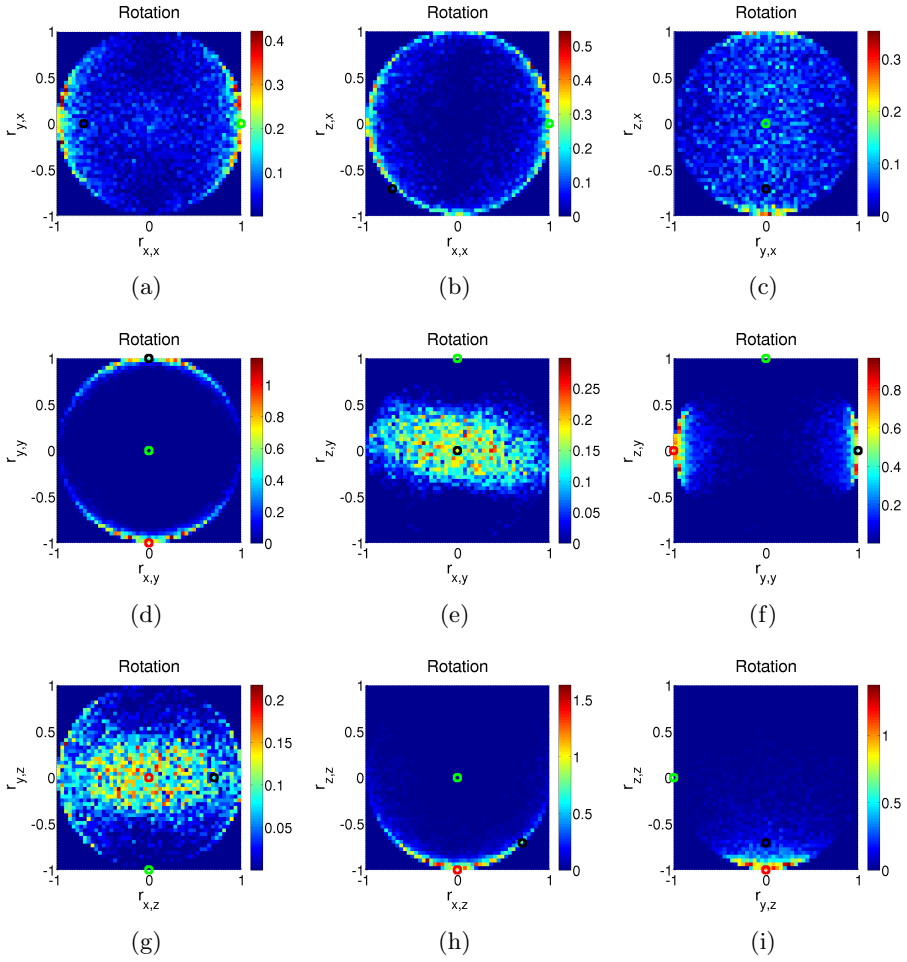
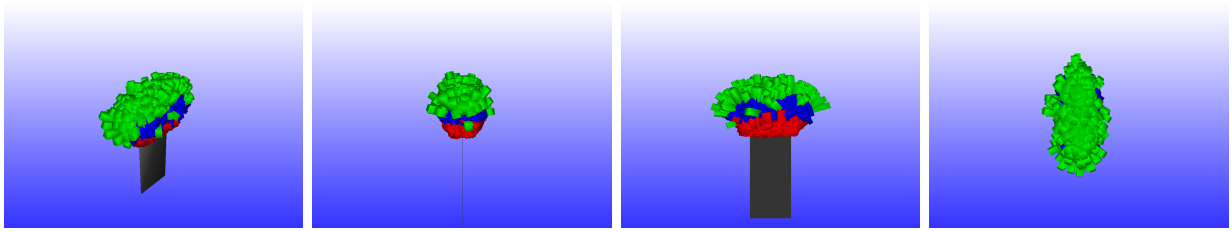


Figure 9: Rotational histograms with three specific rotations for a grasp embedded. Red circle corresponds to $R_{g,1}$, figure 8(a), green circle corresponds to $R_{g,2}$, figure 8(b) and black circle corresponds to $R_{g,3}$, figure 8(c). It should be noticed that the red and green circle is in the same position in the figures 9(a), 9(b) and 9(a), which means the red circle is hidden by the green.

4.2.2 Contour on edge of surface

A sampled version of the simulations results shown in RobWork can be seen in figure 10 from different views. The illustration gives an initial understanding of the distribution of the grasps.



(a) (b) (c) (d)

Figure 10: Visualization with a gripper of 500 successful grasp samples for the contour on surface edge situation. (a) in perspective, (b) side view (ZY), (c) side view (XZ) and (d) top view (XY).

In figure 11 histograms of the distribution in position as well as in orientation are presented. In the positional histogram of x and y the outline of the contour can be seen. In both histograms with x position it can be seen that the grasps are congested at the contour ends and uniformly distributed along the mid of the contour. The x- and z-position plots outlines the contour as well and shows also how the grasps are congested at the ends of the contour.

The rotation of the transformation is analysed axis-wise in the following.

- X-direction: Derived from figure 11(c) the gripper x-axis is primarily directed along the feature x and z-axis corresponding to a rotation around the feature y-axis. Secondary the gripper has some rotation around z-axis of the feature as seen in figure 11(b).

- Y-direction: In figure 11(h) it can be seen that the gripper rotates very little around the x axis of the feature.

- Z-direction: Shows that the gripper z-axis always is directed opposite to the feature z-axis, figure 11(k) , furthermore it can be seen that there is slight rotation around the feature x-axis, figure 11(l) whereas it rotates from 180 degrees around the feature y-axis, figure 11(k).

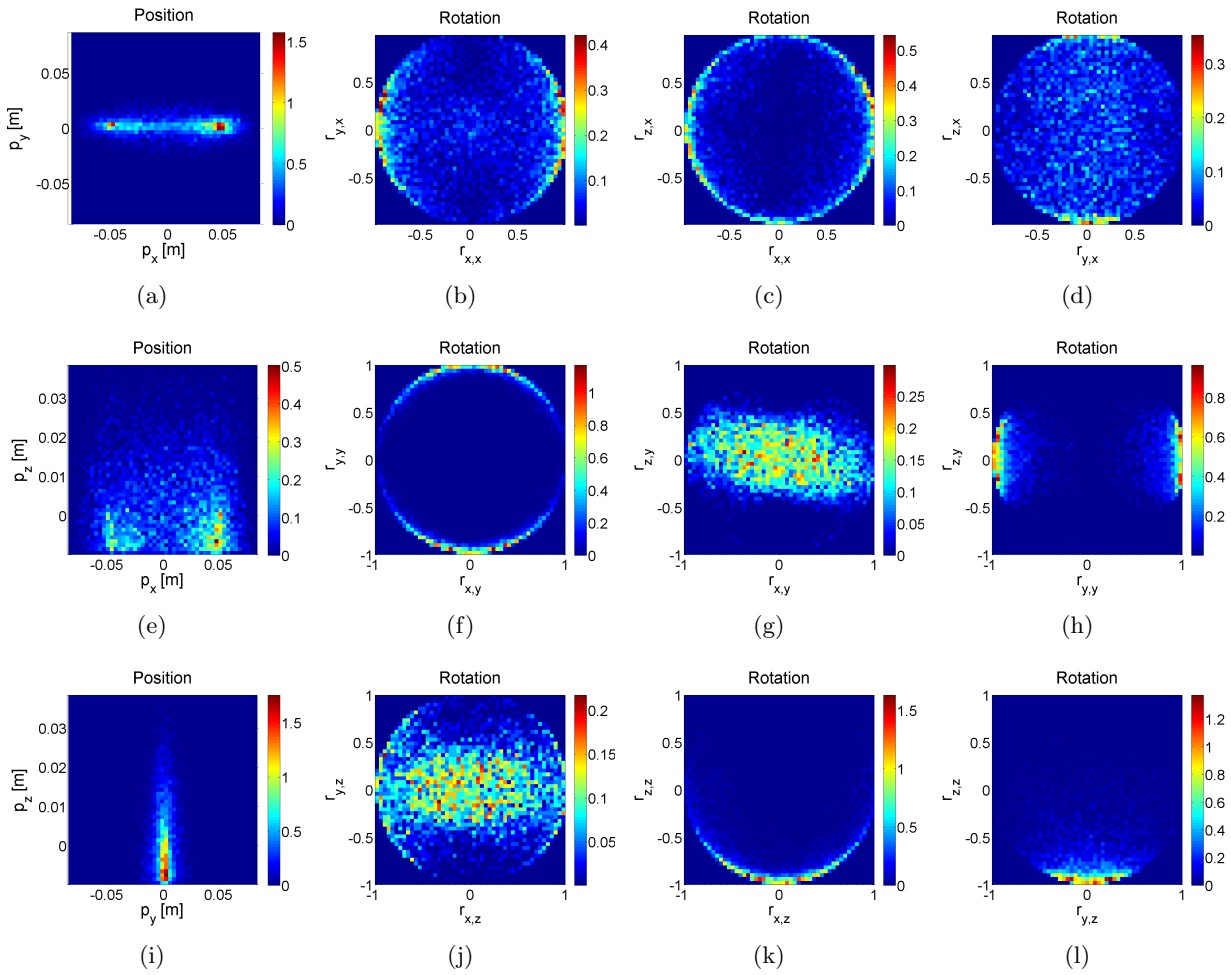
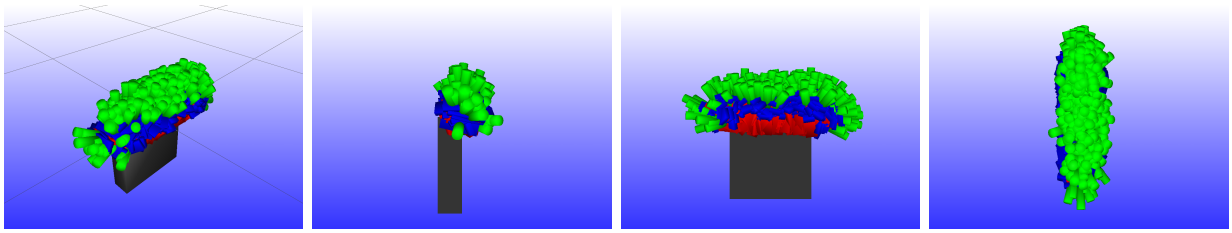


Figure 11: Distribution of the successful grasps in percent for the contour on surface edge situation, given positional histograms in 11(a), 11(e) and 11(i). The remainder of the histograms show the distribution of the rotation of the grasps.

4.2.3 Contour on a box corner

A sampled version of the simulations results shown in RobWork can be seen in figure 12 from different views. The illustration gives an initial understanding of the distribution of the grasps.



(a) (b) (c) (d)

Figure 12: Visualization with a gripper of 500 successful grasp samples for the contour on box situation. (a) in perspective. (b) side view (ZY), (c) side view (XZ) and (d) top view (XY).

In figure 13 histograms of the distribution in position as well as in orientation are presented. The positional part shows 13(a), 13(e) and 13(i) shows that the grasps are distributed along the x-axis, with higher density over the object being in negative y-position. The x- and z-position plot shows a rather uniform distribution of grasp along the contour.

The rotation of the transformation is analysed axis-wise in the following.

- X-direction: In figures 13(b), 13(c) and 13(d), it can be extracted that the gripper x-axis primarily is directed along the feature x-axis, along with the contour and that it rotates little around the z-axis and more around the y-axis.

- Y-direction: It can be seen from figures 13(f), 13(g) and 13(h), that the gripper y-axis is primarily directed along the negative feature y-axis, with some rotation around the feature z-axis.

- Z-direction: It can be seen that the gripper z-axis is directed opposite to the feature z-axis in figure 13(k) and 13(l). In the same figure it can be seen that the gripper can rotate a lot around the feature y-axis whereas it is constrained to a small rotation around the feature x-axis.

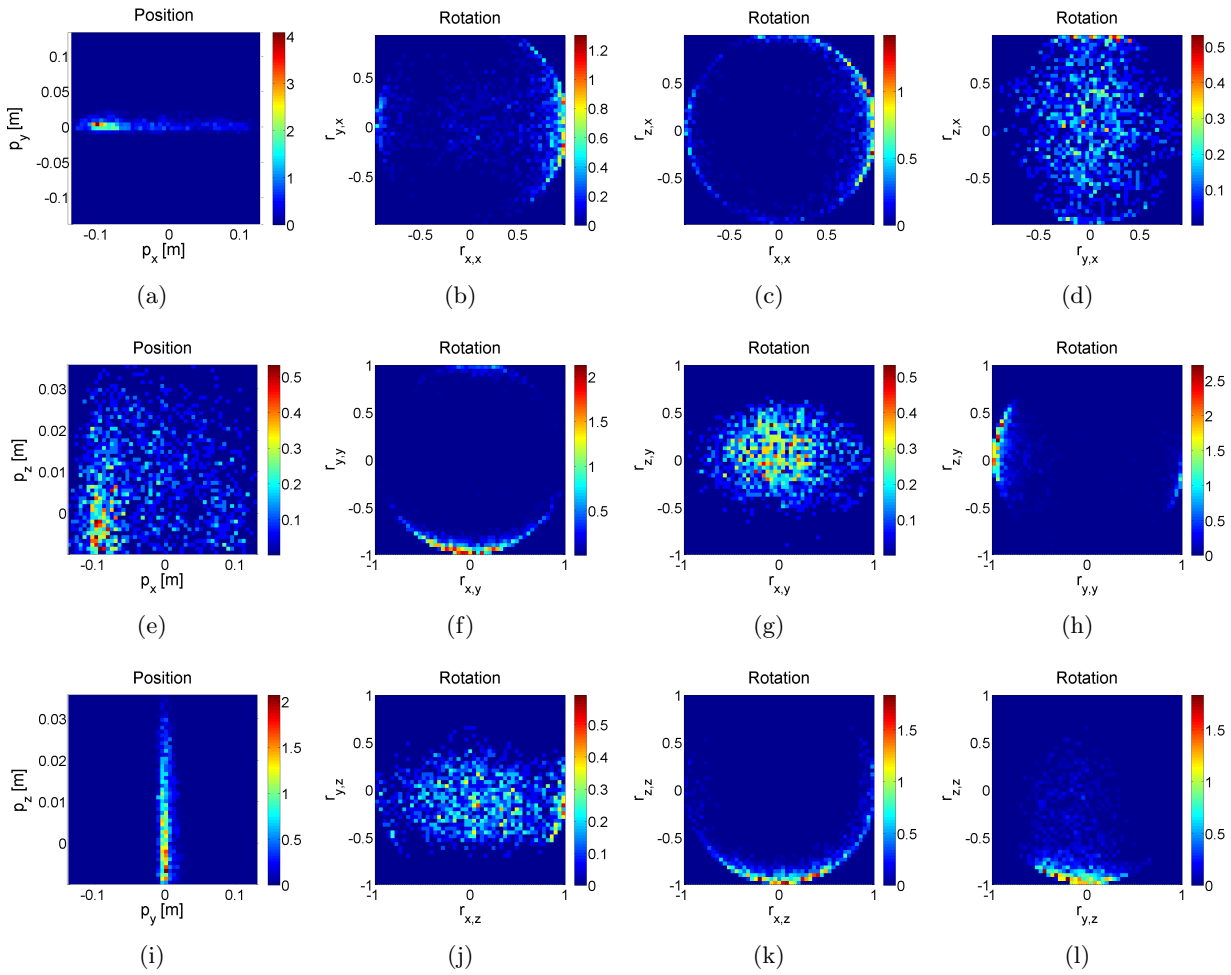
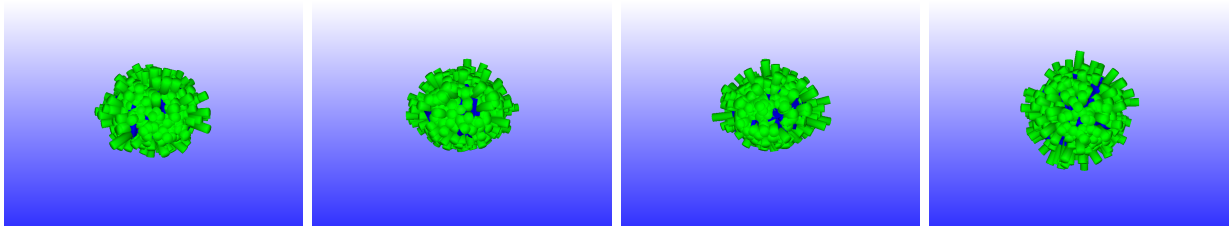


Figure 13: Distribution of the successful grasps in percent for the contour on box situation, given positional histograms in 13(a), 13(e) and 13(i). The remainder of the histograms show the distribution of the rotation of the grasps.

4.2.4 Small surface

A sampled version of the simulations results shown in RobWork can be seen in figure 14 from different views. The illustration gives an initial understanding of the distribution of the grasps.



(a) (b) (c) (d)

Figure 14: Visualization with a gripper of 500 successful grasp samples for for the small surface situation. (a) in perspective. (b) side view (ZY), (c) side view (XZ) and (d) top view (XY).

In figure 15 histograms of the distribution in position as well as in orientation are presented. The positional histograms x and y direction shows the outline of the feature and that the grasps are uniformly distributed on the object. The same is observed when looking at the z position with a slight half sphere shape.

The rotational histograms shows a reasonable uniform distribution with a little weight towards that the gripper z-axis is directed opposite to the feature z-axis.

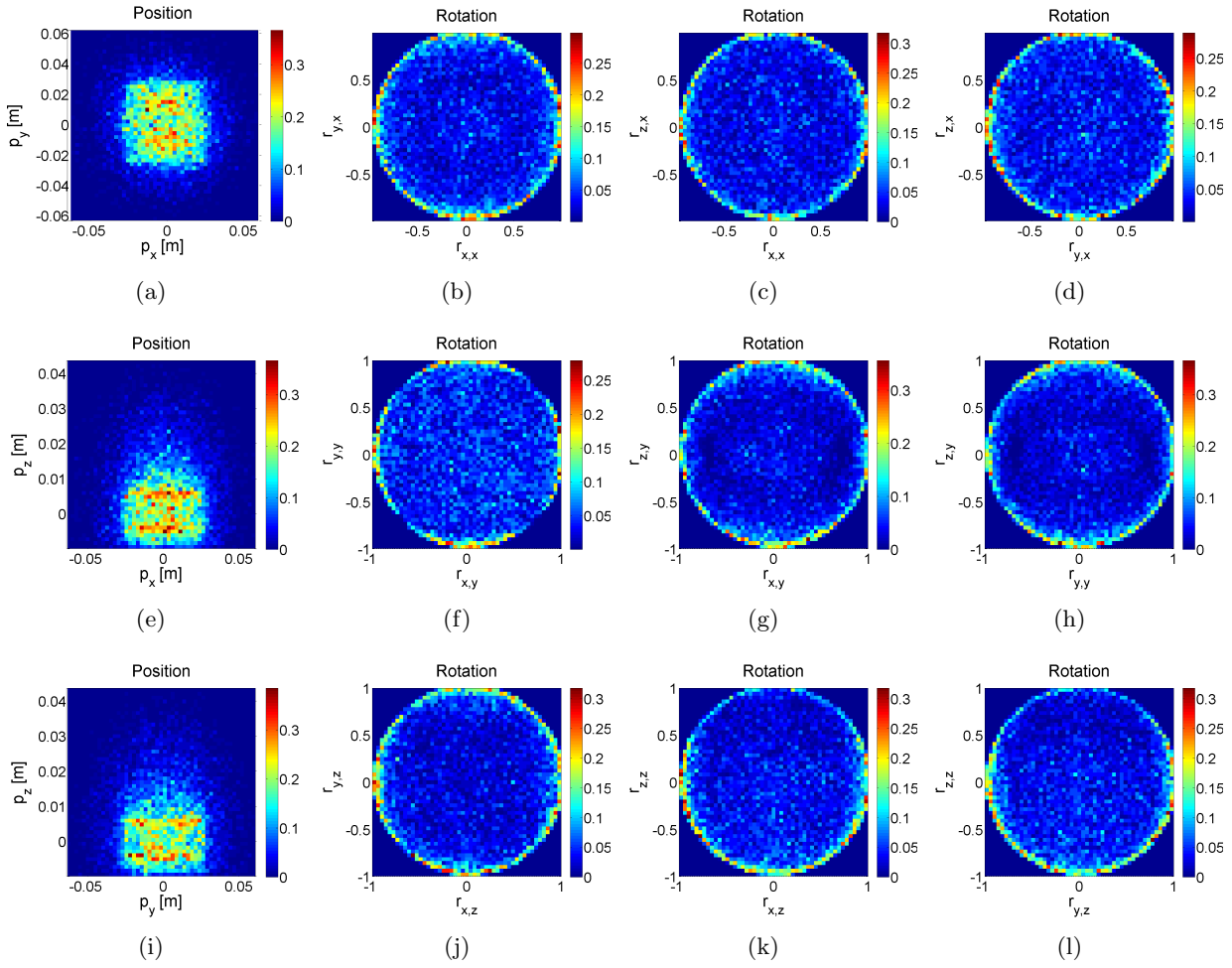
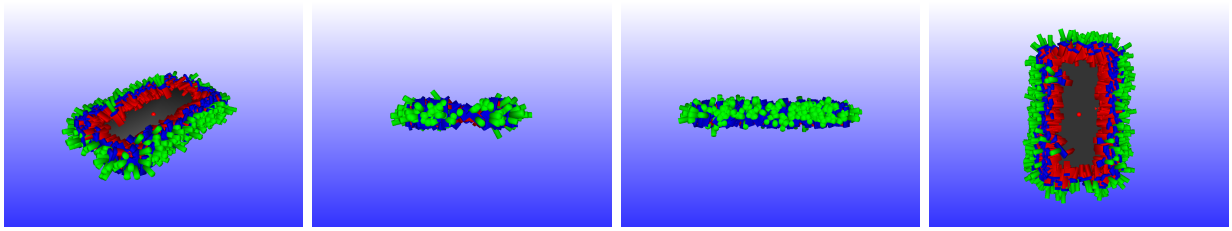


Figure 15: Distribution of the successful grasps in percent for the small surface situation, given positional histograms in 15(a), 15(e) and 15(i). The remainder of the histograms show the distribution of the rotation of the grasps.

4.2.5 Large surface

A sampled version of the simulations results shown in RobWork can be seen in figure 18 from different views. The illustration gives an initial understanding of the distribution of the grasps. It is observed that the grasp are distributed around the edge of the surface.



(a) (b) (c) (d)

Figure 16: Visualization with a gripper of 500 successful grasp samples for the large surface situation situation. (a) in perspective. (b) side view (ZY), (c) side view (XZ) and (d) top view (XY).

In figure 17 histograms of the distribution in position as well as in orientation are presented. The positional part in x and y shows that the grasp are laying along the edge of the surface. The x- and z-position histogram shows that a rather uniform distribution , whereas the y- and z-position shows that the majority of the grasp are along the x-axis.

The rotation of the transformation is analysed axis-wise in the following.

- X-direction: The x-direction is in general aligned with the objects x-axis but with a rotation around the y-axis.

- Y-direction: The y-direction shows that the gripper is aligned with the positive or negative z-axis of the feature.

- Z-direction: The z-direction shows that the gripper is aligned with either the x- or y-axis of the feature.

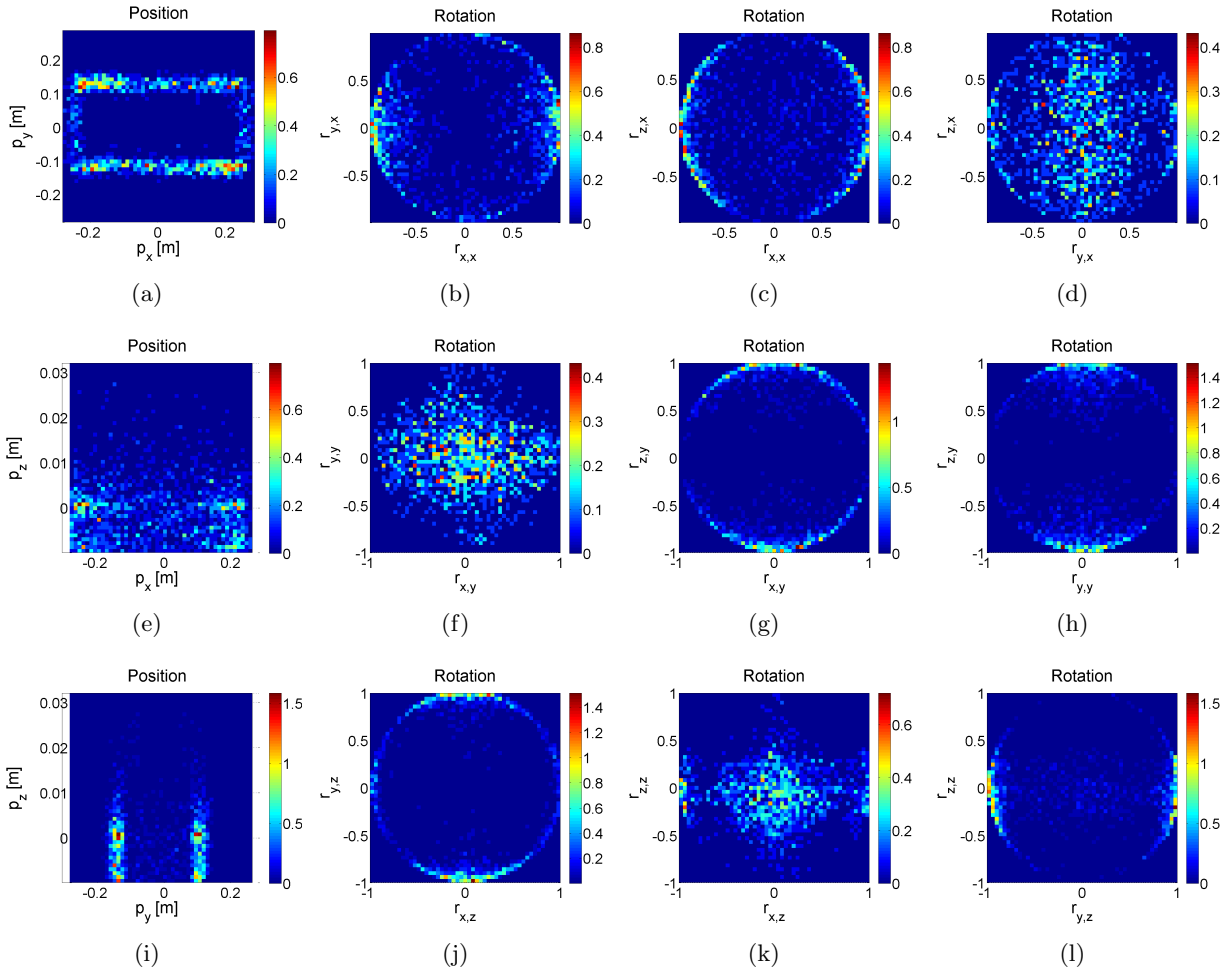
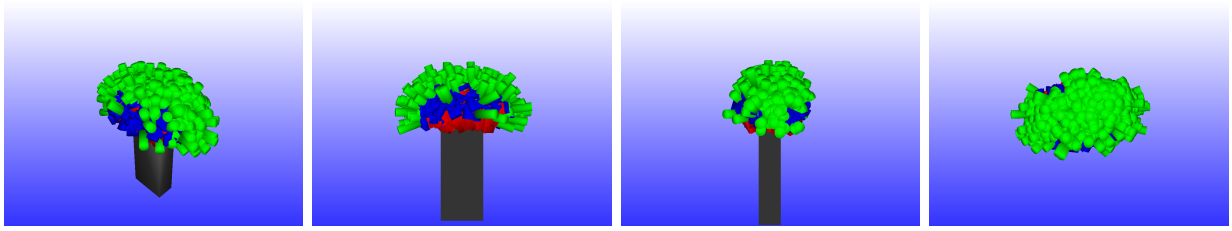


Figure 17: Distribution of the successful grasps in percent for the large surface situation, given positional histograms in 17(a), 17(e) and 17(i). The remainder of the histograms show the distribution of the rotation of the grasps.

4.2.6 Surface on box

A sampled version of the simulations results shown in RobWork can be seen in figure 18 from different views. The illustration gives an initial understanding of the distribution of the grasps.



(a) (b) (c) (d)

Figure 18: Visualization with a gripper of 500 successful grasp samples for the surface on box situation. (a) in perspective. (b) side view (ZY), (c) side view (XZ) and (d) top view (XY).

In figure 19 histograms of the distribution in position as well as in orientation are presented. The positional part in x and y shows that the grasp rather uniformly with some higher densities at two corners. The x- and z-position histogram shows that a rather uniform distribution with higher distribution at the surface edges , whereas the y- and z-position shows a similar picture. The rotation of the transformation is analysed axis-wise in the following.

- X-direction: The x-axis is directed towards the y-axis but are reasonable uniformly distributed.
- Y-direction: The y-axis is primarily directed towards the x axis with slight rotation around y and a larger rotation around the z-axis.
- Z-direction: The z-direction shows that the grasps are distributed with a large rotation around the x-axis and a small rotation around the y-axis, and directed along the negative z-direction.

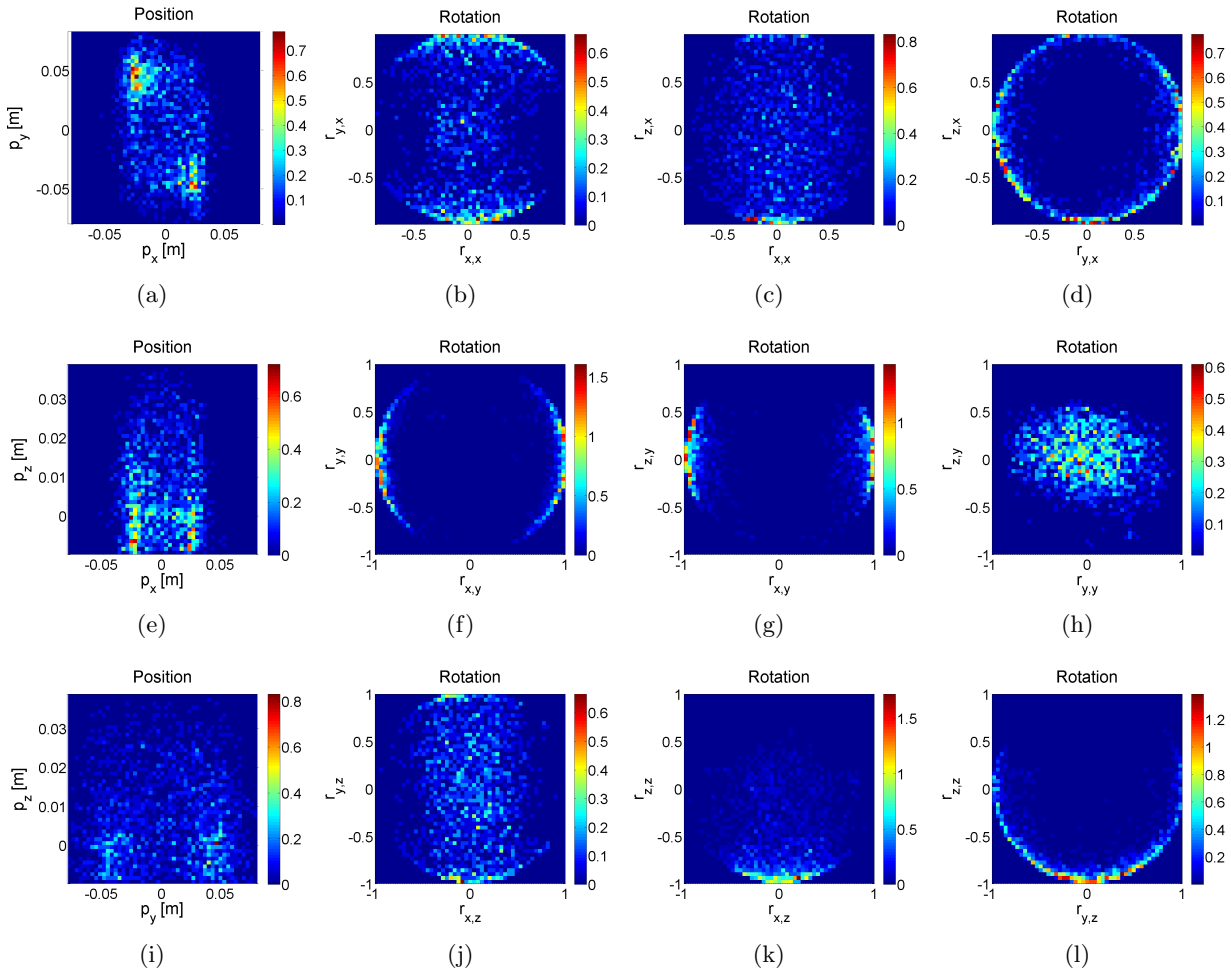
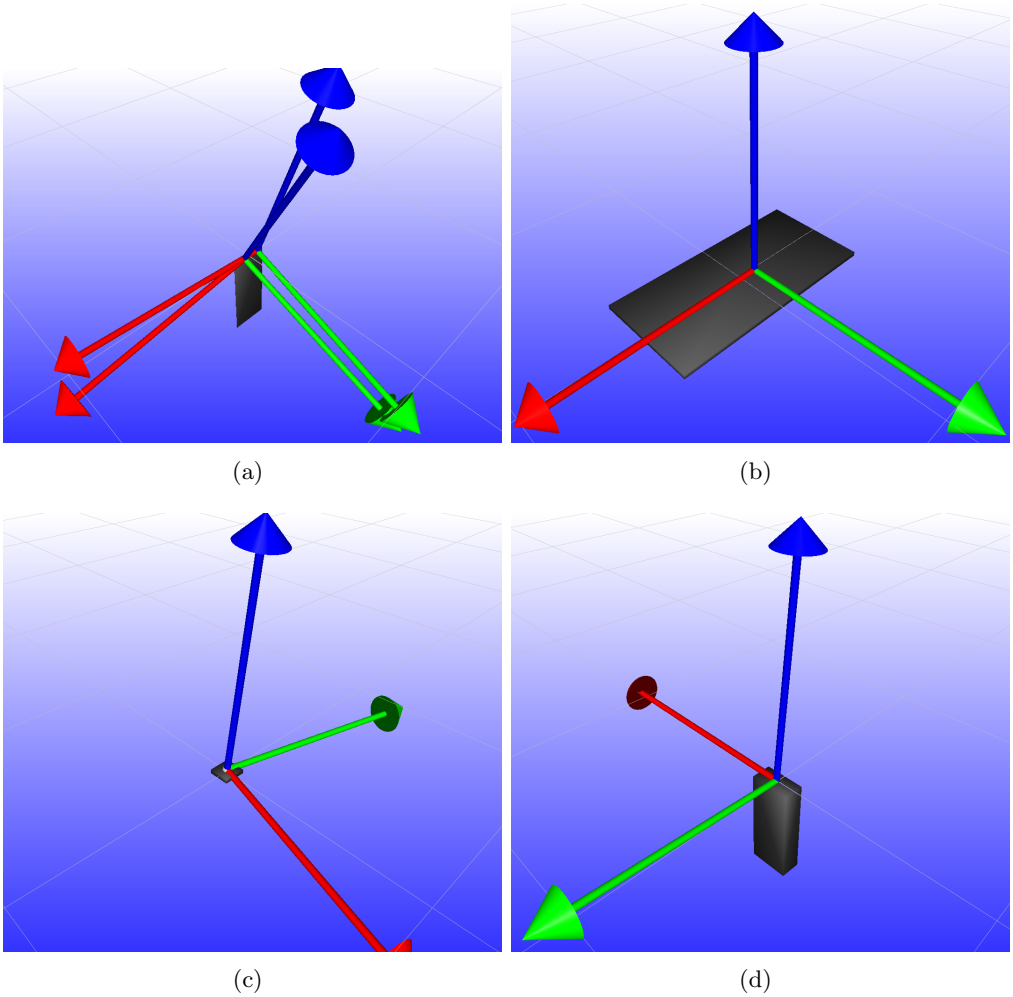


Figure 19: Distribution of the successful grasps in percent for the surface on box situation, given positional histograms in 19(a), 19(e) and 19(i). The remainder of the histograms show the distribution of the rotation of the grasps.

4.3 ECV extraction vs. ground truth

In addition to the ground truth experiments the ECV extracted features are also used. Given a set of stereo images the high level features of 3D surfaces and 3D contours are extracted. From the set of features the corresponding feature of interest is selected and used as reference in the ECV extracted results.

The problem with the extracted features are that they are sensitive to different aspects regarding scene setup, camera position etc. This fact meant for instance that the contour used in the contour on box situation where not found using the chosen camera view, and hence it could not be used as reference frame. In figure 20 the extracted feature reference frame are shown in a RobWork scene with the corresponding feature.



(a) (b)

(c) (d)

Figure 20: ECV extracted feature reference frame vizualized in RobWork, 20(a) for the contour on surface edge situation, 20(b) for the large surface situation, 20(c) for the small surface situation and 20(d) for the surface on box situation. One should notice that the contour in the contour on surface edge situation resulted in two extracted contours.

4.3.1 Contour situations

In the situation containing a contour on a surface edge two contours were extracted hence two results are used as reference. The resulting histograms can be seen in figure 21 and 22. The plots shows that the positional parts is off due to the surface has been split into two and the rotational part suffers from the problem that the z- and y-axis is not aligned with the ground truth.

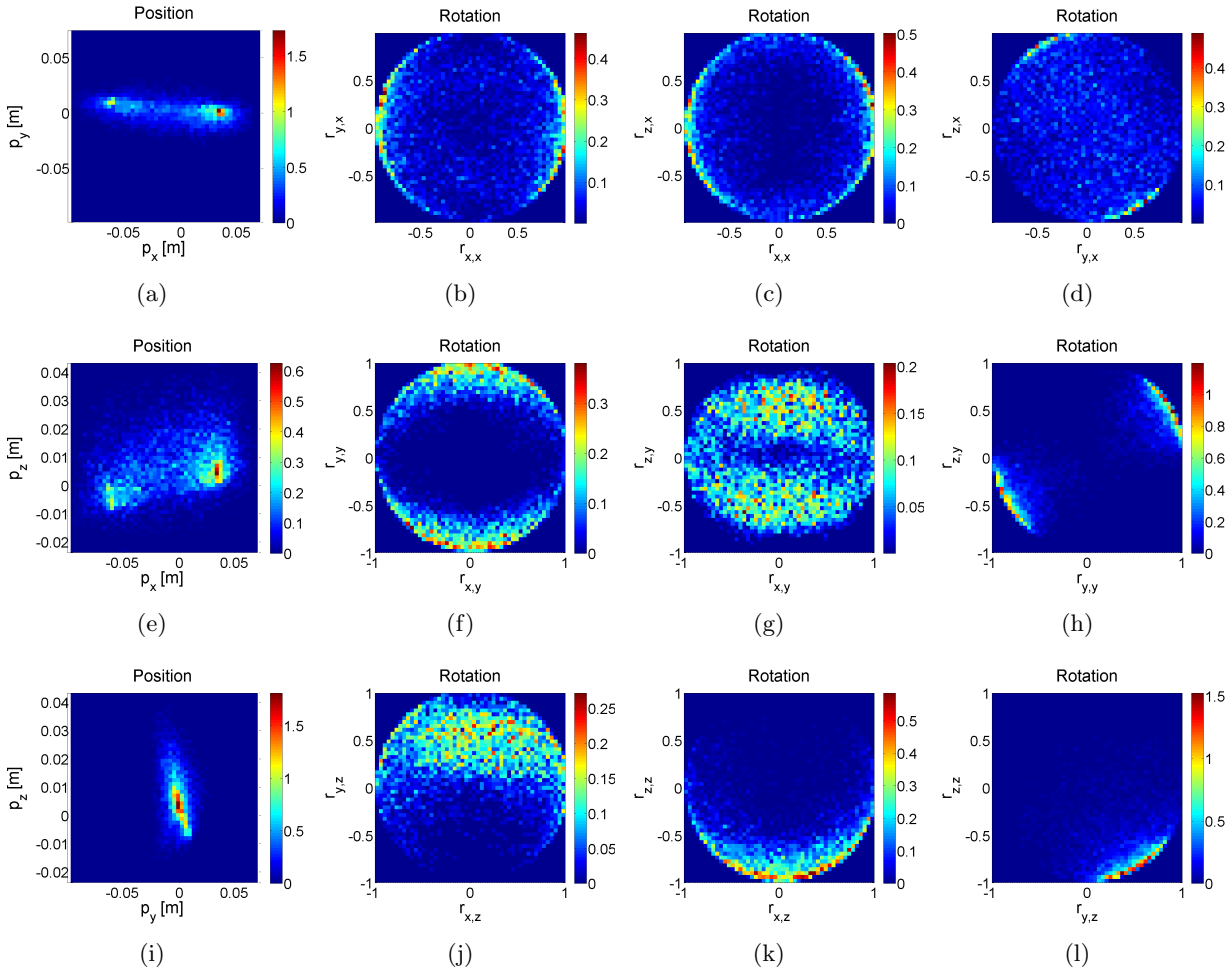


Figure 21: Results of the feature to grasp association for contour on surface edge situation, when associated with the first ECV extracted contour feature.

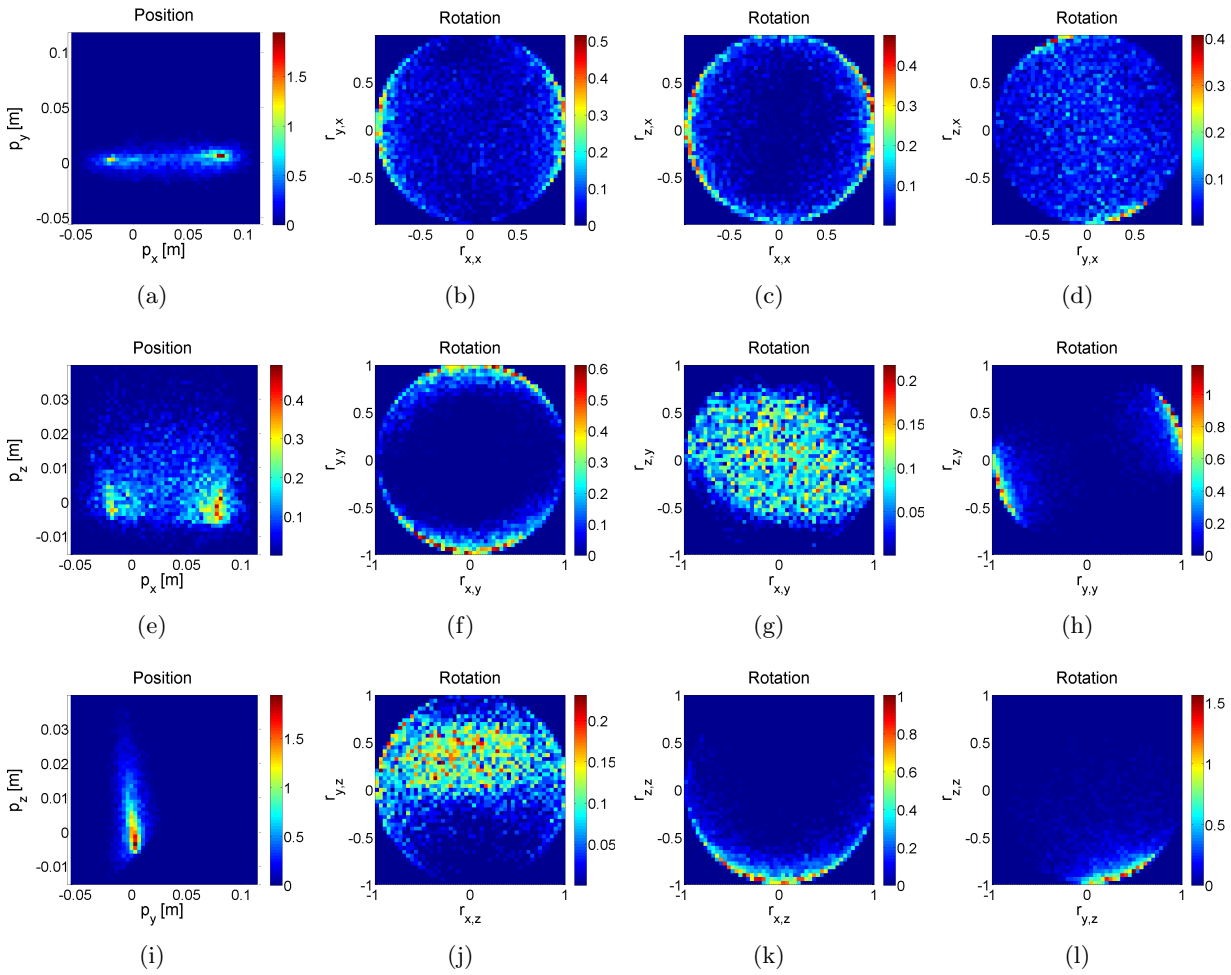


Figure 22: Results of the feature to grasp association for contour on surface edge situation, when associated with the second ECV extracted contour feature.

4.3.2 Surface situations

In general the surface situation seems to yield comparable results to the ground truth reference. In figure 23 the feature-grasp association for the small surface with respect to a ECV extracted surface feature is shown. The ECV extracted surface feature of the small surface is slightly off in the orientation, which can be seen in the skew that the positional histogram shows. In the rotational part this is not noticeable because of the uniform distribution.

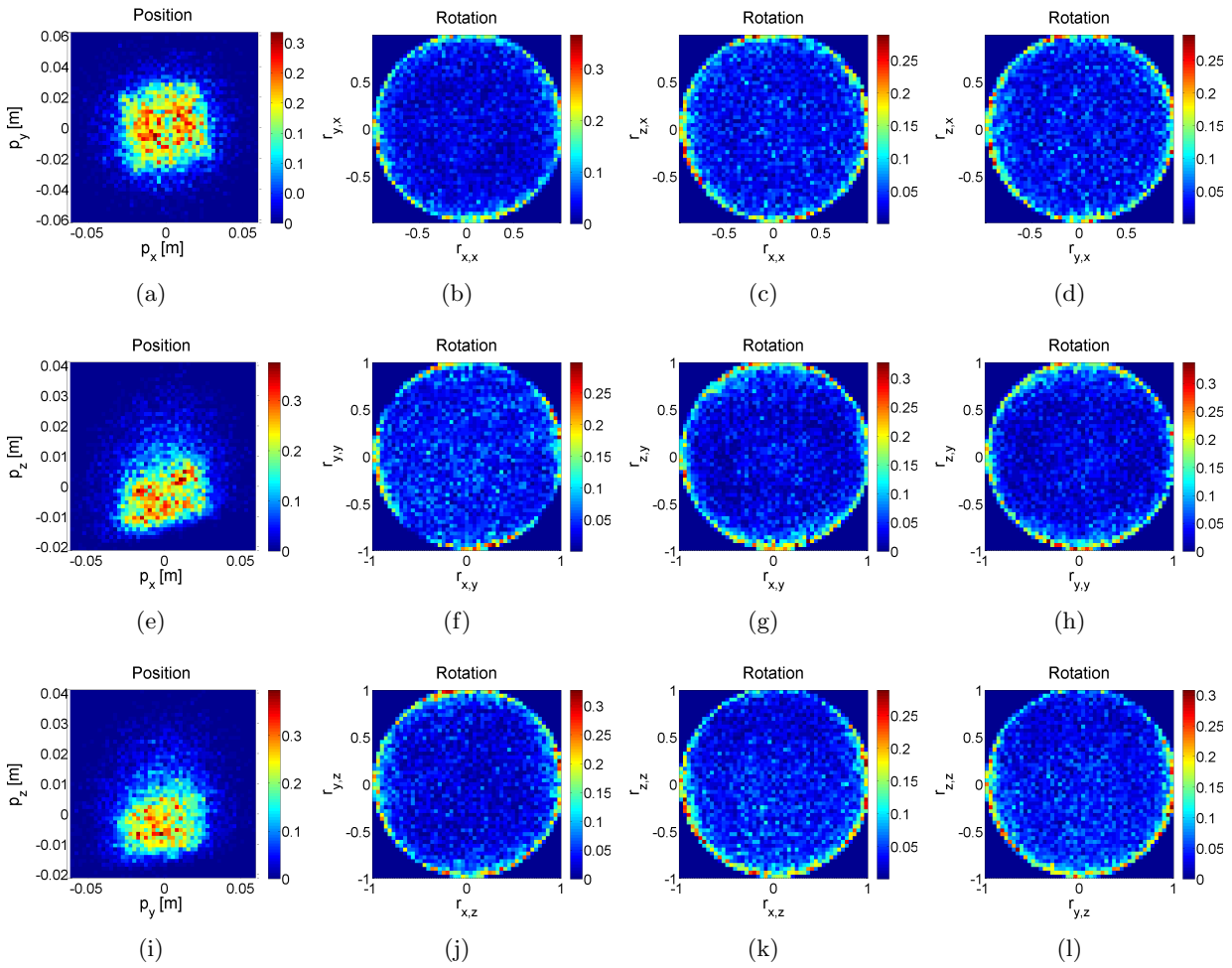


Figure 23: Results of the feature to grasp association for small surface situation, when associated with ECV extracted surface feature.

In figure 24 the large surface situation with reference to the extracted surface feature can be seen. The histogram is almost identical to the ground truth histograms and for this reason no further remarks considering it.

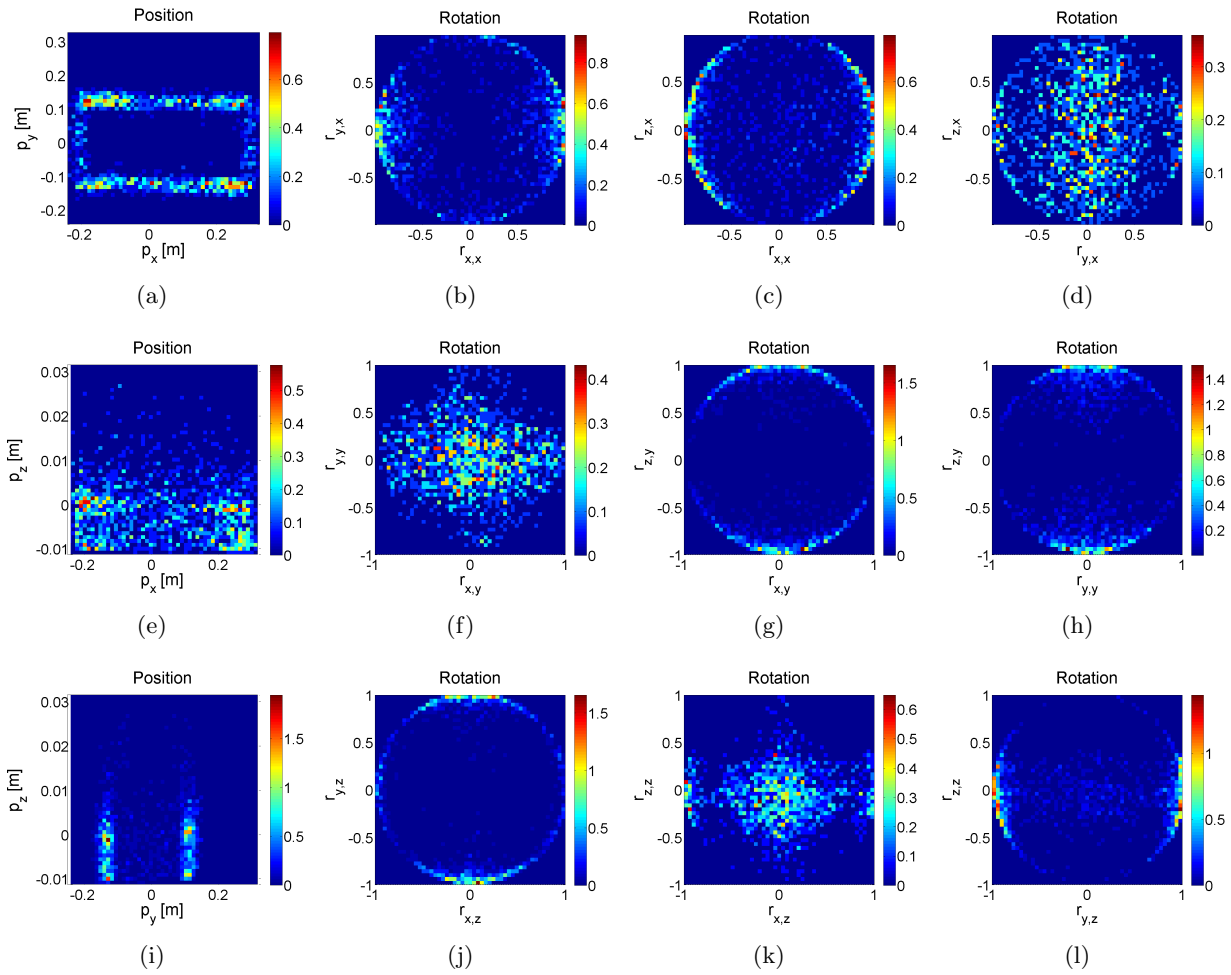


Figure 24: Results of the feature to grasp association for large surface situation, when associated with ECV extracted surface feature.

In figure 25 the histogram for the extracted surface feature in the surface on box situation is shown. As with the large surface situation the extraction yields almost the same frame as the ground truth despite a switched axis.

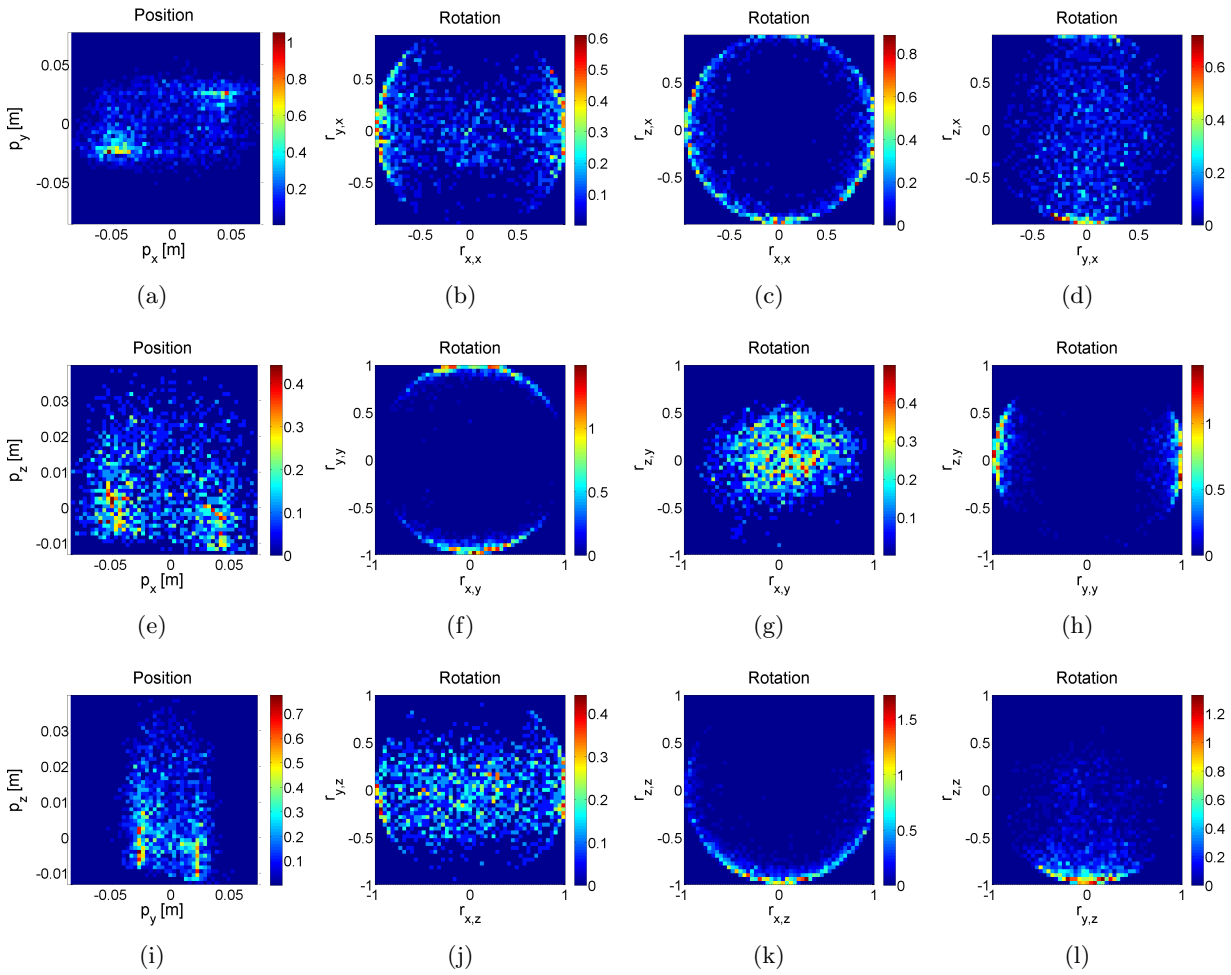


Figure 25: Results of the feature to grasp association for surface on box situation, when associated with ECV extracted surface feature.

5 Discussion

Different aspects of the acquired results are discussed in the following subsections respectively for contour situations and surface situations.

5.1 Contour situations

The qualitative analysis of the feature to grasp association for contour features showed that the contours in different situations resulted in different grasp spaces. The grasp spaces were explained in terms of rotation as well as position.

5.1.1 ECV extracted contours

In one of the situations the contour of interest was not extracted. This exposes the problem were a specific scene setup with camera and object placement results in a missed feature during the

extraction process. This however can be solved by having multiple camera views, which covers the scene in an sensible way.

Another problem for the contour is that the orientation is not unambiguously defined. The problem with the ECV contours is that the orientation of a contour is derived from a principal component analysis, which finds the three main axis for the sub feature cloud. An ideal straight line contour however would only have one principal axis namely along the contour, this means that it is not possible to find the other axes, and hence are the extracted principal axis of the contour not useful for a generalization. To compensate for the inadequacy in the contour representation it is proposed to introduce higher order feature relation between multiple features. Given high order relation it is suggested that a generalization of the feature to grasp space for specific feature sets are more likely to be found and hence be applied to predict the probability of successful grasps.

5.2 Surface situations

The surface situations and corresponding qualitative analysis of the results showed in general what was expected, in particular that the gripper were directed opposite to the feature reference frame and that the dimensions of the objects limited the distribution of grasps.

The small surface situation shows that just about any grasp would succeed which was expected due to the dimension of the surface and the gripper. The high success-rate was also seen in the simulation statistics were it hit around 20 % compared to 1 to 8 % for the other objects. In the large surface situation the results shows that the successful grasp is spread around the edge of the surface which was expected in connection with gripper dimensions.

The surface on the box situations showed expected results such that gripper were able to grasp a 180 degrees rotations around the feature y-axis and only slightly around the feature x-axis. One noticeable thing is how similar the feature to grasp space for the surface on box situation, figure 19 and the contour on surface edge, figure 11 looks in particular in the orientation. This is explained as the contour being a very narrow surface or the surface to be a very wide contour.

5.2.1 ECV extracted surfaces

The extracted surface features turned out very successful with the only problem being the pose of the surface in the small surface situation, this however can to some extent be explained by the rather small size of surface. Given a small surface consist if less sub features and hence the surface extraction would risk relying more on a badly extracted sub feature. For the larger surfaces in the other surface situations the extracted and ground truth histograms where very similar which talks in favour of a generalization of the grasp to feature relation.

As with the contour situation it is proposed that by utilising multiple features and their relation a more reliable feature to grasp space can be established, which is not affected of a single feature being inadequate.

5.3 Relation to previous work

When examining the results it can be derived that the grasps which were predefined in [5] would yield a good probability of a successful grasp given that the features occur in the situations shown in this context. This translates to the following statement. Given that the feature situation in this

paper are reasonable aligned with situations for real objects the predefined simple grasp are shown to be sensible. This could be proven by analysing the feature relations in the objects used for the evaluation of the simple grasps.

6 Conclusion

Throughout this journal the feature to grasp association has been investigated in terms of ECV high level features of 3D contours and 3D surfaces in a set of different situation designed to exploit different parts of the feature to grasp space.

By grasp simulation using RobWorkSim the different situations have been evaluated by uniformly sampling grasps in the vicinity of the feature of interest.

Given the feature reference frame and the homogeneous transformation to the successful grasps from simulation, the feature to grasp association space is spanned, initially in terms of a ground truth feature reference frame and afterwards with respect to a visually extracted ECV feature reference frame. Each of the different situations have been qualitatively analysed by explanation of the significant structures in the feature grasp space. Hereafter the resulting feature to grasp space were evaluated in terms of a ECV extracted feature reference frame.

The comparison of the results, for respectively ground truth feature reference frame and the extracted feature reference frame, unveiled however a number of different problems, which relates to two different aspects namely the extraction process and the feature representation. The extraction of the contours turned out difficult as one of the proposed contours were not extracted and the other one were divided into two contours. The surface extraction however showed reasonable results. The problem of feature representation were also revealed in the comparison as the ambiguity in the contour representation were found.

Using the feature extraction for a single camera view, it was realized that the feature extraction process has some built in difficulties, meaning that the extraction is not always giving the expected results. To compensate or increase the likelihood of a reliable feature extraction, it is proposed that multiple camera views should be the foundation for the extraction, thereby reducing the effect of one difficult camera view to the extraction quality.

The single feature problem can be condensed to the statement that a single feature does not carry enough information to give a reliable grasp space. In order to overcome the inadequate information for a single feature, it is proposed that an introduction of multiple features and the relation between them will compensate for this, essentially meaning that combination of a set of features would provide the necessary information.

Finally the results were related to the work presented in [5] explaining the success of simple predefined grasps in connection of the 3D contours and 3D surfaces.

References

[1] E. Başeski, N. Pugeault, S. Kalkan, D. Kraft, F. Wörgötter, and N. Krüger. A scene representation based on multi-modal 2d and 3d features. *3D Representation for Recognition Workshop (in conjunction with ICCV)*, 2007.

[2] Lars-Peter Ellekilde and Jimmy Alison Jorgensen. RobWork: A Flexible Toolbox for Robotics Research and Education. 2010.

[3] Jimmy A. Jørgensen, Lars-Peter Ellekilde, and Henrik G. Petersen. Robworksim - an open simulator for sensor based grasping. *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1 –8, june 2010.

[4] Mila Popović, Gert Kootstra, Jimmy Alison Jørgensen, Danica Kragic, and Norbert Krüger. Grasping unknown objects using an early cognitive vision system for general scene understanding. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 987–994. IEEE, 2011.

[5] Mila Popović, Dirk Kraft, Leon Bodenhagen, Emre Başeski, Nicolas Pugeault, Danica Kragic, Tamim Asfour, and Norbert Krüger. A strategy for grasping unknown objects based on coplanarity and colour information. *Robotics and Autonomous Systems*, 58(5):551 – 565, 2010.

[6] Schunk. Datasheet: Schunk PG70, http://www.schunk.com/schunk_files/attachments/pg_70_en.pdf, Januar 2012.

Integrating visual processing and manipulation for autonomous learning of object representations

Aleš Ude, David Schiebener, Hirokazu Sugimoto, and Jun Morimoto

***Abstract*— Learning about new objects that a robot sees for the first time is a difficult problem because it is not clear how to define the concept of object in general terms. In this paper we consider as objects those physical entities that are comprised of features which move consistently when the robot acts upon them. Among the possible actions that a robot could apply to a hypothetical object, pushing seems to be the most suitable one due to its relative simplicity and general applicability. We propose a methodology to generate and apply pushing actions to hypothetical objects. A probing push causes visual features to move, which enables the robot to either confirm or reject the initial hypothesis about existence of the object. Furthermore, the robot can discriminate the object from the background and accumulate visual features that are useful for training of state of the art statistical classifiers such as bag of features.**

I. INTRODUCTION

Statistical approaches to object recognition and categorization have received a lot of attention by the computer vision community in recent years. Excellent performance and state-of-the-art results have been achieved with methods such as bag-of-features, which represent an image as a collection of local feature points [2], [24]. However, the bag-of-features methods do not have a built-in ability to segment objects from the background [12]. This can significantly reduce the performance of object recognition, especially if the object image covers only a small portion of the whole image. Designing a reliable and general object segmentation system that works in many different environments and under varying lighting conditions is an extremely difficult problem, but is a necessary component of an autonomous robot. While statistical learning can overcome some of these problems, it typically requires the robot to acquire and process many training images. This is not an option for an autonomous robot, which needs to have the ability to expand its library of objects as quickly as possible to be able to operate in unstructured and uncontrolled environments.

In this paper we propose to overcome the problem of identifying and learning new objects by exploiting the manipulation capabilities of a humanoid robot like the one in Fig. 1. If object manipulability is taken into account, it is much easier to define the concept of object than when only visual characteristics are used [3]. Based on the concept of object manipulability, we can define objects as physical entities that are manipulable by the robot and whose features move consistently when the robot manipulates them. Such characteristics were also used by Gibson [7] to define the concept of object and were exploited for figure-ground segmentation in a number of previous works [5], [10], [11], [13], [20], [21]. While some of these works assume that the object has been first grasped [10], [11], [21], others do allow for simpler actions such as pushing [5], [13], [20] (also called poking, nudging). Although pushing results in a less controlled motion of the pushed object than manipulation after grasping, probing pushing actions are much easier to generate than actions that include grasping.

In this paper we present a new methodology to generate probing pushes necessary to confirm or reject the initial object hypotheses and techniques for segmenting and learning of unknown objects. Based on 3-D points obtained from local features, regular surface patches and point clusters are detected to form initial object hypotheses. These hypotheses are then validated by the robot as it attempts to push the hypothetical objects. We utilize linear, autonomous dynamic systems to generate the probing pushes. The induced motion provides sufficient cues for distinguishing the pushed object from its environment. After the existence of an object has been confirmed, it is pushed repeatedly to segment and accumulate the features that move in unison with it. We demonstrate that these features enable reliable object learning and recognition. The developed method requires no prior knowledge about the object or the environment, the only

Fig. 1. Humanoid robot CB-i touching an object placed on the table. It has an active visual system, which on the one hand improves the object fixation capabilities, but on the other hand reduces the accuracy of 3-D vision.

A. Ude and D. Schiebener are with Jožef Stefan Institute, Department of Automatics, Biocybernetics, and Robotics, Jamova 39, Ljubljana, Slovenia ales.ude@ijs.si, david.schiebener@ijs.si

J. Morimoto H. Sugimoto, and A. Ude are with ATR Computational Neuroscience Laboratories, Department of Brain Robot Interface, Kyoto, Japan xmorimo@atr.jp, aude@atr.jp

Fig. 2. Initial hypotheses that were generated for a number of typical household objects. Crosses of the same color belong to the same hypothesis.

necessary assumptions are that the object contains some distinctive visual features and moves as a rigid body.

II. Searching for Objects

For the generation of initial object hypotheses we use visual information obtained from stereo cameras of the humanoid robot. In particular, we determine 3-D points within the field of view using stereo calibration on an active camera system [22]. Like in our previous work [20], we use the Harris interest point detector [8] to find points that allow robust stereo matching. These salient points are mostly located in highly textured parts of the image. In our current system, we additionally use color-based maximally stable extremal regions (MSER) [15], [6] as a second type of interest points to complement the Harris interest points in image regions with less texture.

For both Harris interest points and color MSERs, we perform stereo matching using epipolar geometry. In this way we obtain a set of 3-D points, which are usually very reliable and accurate when calculated from Harris interest points, but somewhat less precise, although still mostly useful, when determined from color MSERs. If an object has large untextured areas on its surface, the hypothesis generation benefits significantly from the use of MSERs in addition to the Harris interest points, as can be seen in Fig. 3.

Amongst these 3-D points we look for possible objects. The criteria we use for the initial calculation of object hypotheses are smoothness of surface patches and local proximity of subsets of the detected points. As we consider smooth surface patches to be a more reliable hint about the underlying structure, we search for them first. Planar, spherical and cylindrical surface patches are detected amongst the points using RANSAC [4]. This algorithm repeatedly chooses a random subset of 3-D feature points,

Fig. 3. The left image shows hypotheses generated using only Harris interest points. In the right image, color MSERs are used in addition to them, which enables us to recover the surfaces more completely.

calculates the parameters of the considered kind of surface from them, counts how many points of the overall set lie within a tolerance of that surface, and returns the best found parameters. It is a robust statistical method and is therefore well suited to detect structures that contain only a small portion of 3-D feature points, which is usually the case in our scenario, especially when there are several objects in the field of view. Details about the detection of planes, spheres and cylinders are given in Sec. II-A and II-B.

From each of the hereby generated hypotheses we remove the points that are far away from the hypothesis' center compared to the extent of the region enclosed by them, as there is a high risk that such feature points are outliers. To avoid subsuming several objects into one hypothesis, we apply X-means [16] to each hypothesis and divide it if that seems appropriate. By doing so, we might create several hypotheses lying on the same object, but that is not a serious problem. All features that belong to the object will later be added again to the hypothesis if they move in unison with it when the object is pushed (see section III-A).

We search for all three considered kinds of surface patches simultaneously and keep the hypothesis that contains the maximum number of feature points, which are then removed from the complete feature point set. This process is repeated with the remaining points until no surface containing more than a minimum number of feature points is found. Finally, we apply X-means clustering algorithm to the remaining points, and the resulting clusters are added as hypotheses if they contain enough points and have a high points-per-volume ratio. With this last step, we can detect objects that contain a cluster of interest points and/or color MSERs, which do not lie on any of the considered smooth surfaces.

A. Planes and spheres

A plane in 3-D space is uniquely defined by three non-collinear points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Its normal $\mathbf{n}$ can be calculated from these three non-collinear points as $\mathbf{n} = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)$. The plane is then given by the equation $\mathbf{n}^T\mathbf{x} + d = 0$, with $d = -\mathbf{n}^T\mathbf{x}_1$. This equation must be fulfilled by all points $\mathbf{x}$ lying on the plane.

A sphere is uniquely defined by four non-coplanar points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. Its parameters can be calculated in a closed form, too. The center $\mathbf{c}$ and radius r of the sphere can be calculated from the points by solving the determinant

equation $|\mathbf{M}| = 0$, where

$$\mathbf{M} = \begin{bmatrix} \mathbf{x}^T\mathbf{x} & \mathbf{x}^T & 1 \\ \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T\mathbf{x}_2 & \mathbf{x}_2^T & 1 \\ \mathbf{x}_3^T\mathbf{x}_3 & \mathbf{x}_3^T & 1 \\ \mathbf{x}_4^T\mathbf{x}_4 & \mathbf{x}_4^T & 1 \end{bmatrix}. \quad (1)$$

Let $\mathbf{M}_{ij}$ denote the submatrix of $\mathbf{M}$ formed by leaving away row i and column j. The solution is given by

$$\mathbf{c} = \begin{bmatrix} 0.5\frac{|\mathbf{M}_{12}|}{|\mathbf{M}_{11}|} \\ -0.5\frac{|\mathbf{M}_{13}|}{|\mathbf{M}_{11}|} \\ 0.5\frac{|\mathbf{M}_{14}|}{|\mathbf{M}_{11}|} \end{bmatrix}, \quad (2)$$

$$r = \mathbf{c}^T\mathbf{c} - \frac{|\mathbf{M}_{15}|}{|\mathbf{M}_{11}|}. \quad (3)$$

If $|\mathbf{M}_{11}| = 0$, the four points are coplanar and there is no solution.

We can find a plane or a sphere in the 3-D point set using RANSAC, where the following steps are repeated N_p times:

- select 3 (4) points at random,
- calculate the parameters of the plane (sphere) defined by these points,
- count how many of the points from the set lie on the plane (sphere).

The plane (sphere) with the maximum number of inliers is then returned.

B. Cylinders

The detection of cylinders within a point set is more complicated because the parameters of a cylinder can not be determined so easily from a few points on its surface. We applied the algorithm proposed in [1], which uses a 2-stage RANSAC approach, first estimating the cylinder axis and then the appropriate radius and offset from the origin of that axis.

Promising candidates for the cylinder axis can be found by analyzing local surface normals. They are calculated from all points and their nearest neighbors and, once normalized, all lie on a unit sphere. A cylinder amongst the point set corresponds to a great circle on the unit sphere. Such great circles are equivalent to the intersection of the sphere with a plane through its origin. Consequently, using RANSAC, the great circle with a maximum number of inliers can be found by testing the great circles defined by the plane through two randomly chosen normals and the origin. The normal of the optimal plane is chosen as the candidate cylinder axis for the next step.

For a given cylinder axis, its offset and the cylinder radius can be determined easily because this problem can be reduced to finding a two dimensional circle. All 3-D points whose local surface normals contributed to the great circle are projected onto the plane orthogonal to the cylinder axis. Using RANSAC again, the circle with the maximum number of points lying on it can be found, exploiting the fact that three non-collinear 2-D points (x_i, y_i) define a circle. Its center coordinates (x_c, y_c) are given by

$$x_c = \frac{(y_3 - y_2)(x_1^2 + y_1^2) + (y_1 - y_3)(x_2^2 + y_2^2)}{2\delta} + \frac{(y_2 - y_1)(x_3^2 + y_3^2)}{2\delta}, \quad (4)$$

$$y_c = \frac{(x_3 - x_2)(x_1^2 + y_1^2) + (x_1 - x_3)(x_2^2 + y_2^2)}{2\delta} + \frac{(x_2 - x_1)(x_3^2 + y_3^2)}{2\delta}, \quad (5)$$

where

$$\delta = x_1(y_3 - y_2) + x_2(y_1 - y_3) + x_3(y_2 - y_1), \quad (6)$$

and the radius is simply the distance of one of these points to the center. The radius of the resulting circle is the radius of the cylinder, and the cylinder axis passes through the center of the circle.

In every iteration of the outer RANSAC loop, a new possible cylinder axis is determined that has to be different from the axes which have already been tested. After a fixed number of iterations, or when no promising new axis can be found anymore, the parameters of the cylinder with the maximum number of inliers are returned.

III. Active Vision and Pushing

The initial object hypothesis includes information about the hypothetical object position, which can be used to generate a probing pushing movement. However, on a robot with many degrees of freedom and an active eye system like in Fig. 1, we cannot rely on the system being accurately calibrated. Even though we account for the eye configurations when calculating stereo triangulation [22], the calculated locations are still rather inaccurate in the robot's body frame. To improve the accuracy of the probing pushing movements, we included a learning component into our system.

Training is done by moving a robot to a number of locations on the table, on which the robot should look for new objects. We place an object that our vision system can easily localize at a location where the robot hand touches it. Thus, we acquire the following data

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, \quad (7)$$

where $\mathbf{x}_i$ is the position of the object as estimated by the robot's visual system and $\mathbf{y}_i$ are the associated joint angles specifying the robot configuration, including the configuration of its eyes. In our experiments we placed the robot arm at N different locations on a regular grid. To avoid the need for using the robot's inverse kinematics, we estimate function

$$\mathbf{F} : \mathbf{x} \mapsto \mathbf{y}, \quad (8)$$

where $\mathbf{x} \in \mathbb{R}^3$, $\mathbf{y} \in \mathbb{R}^D$, and D is the number of robot degrees of freedom relevant for the task. We applied Gaussian process regression (GPR) [18], which is a state-of-the-art statistical function approximation method, to estimate this function. Given a new desired hand position $\mathbf{x}^* \in \mathbb{R}^3$, the training data $\{\mathbf{x}_i, \mathbf{y}_i\}$, and writing $\mathbf{y}^j = [y_1^j, \ldots, y_N^j]^T$,

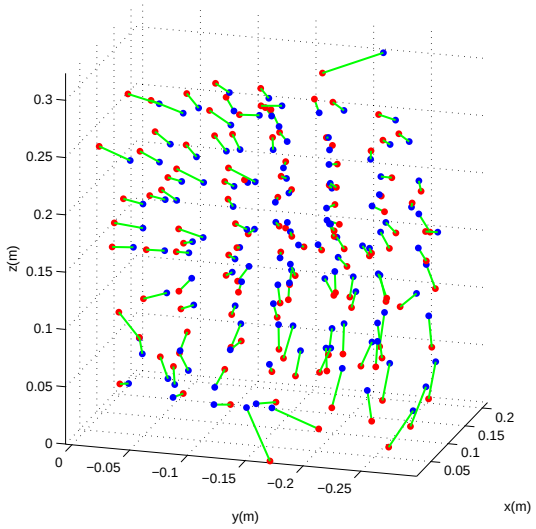


Fig. 4. Blue dots show the robot hand positions estimated by vision, whereas the red dots show the positions calculated by forward kinematics from joint configurations, which were estimated by Gaussian process regression using formula (9). There is a significant systematic error, which is illustrated by green lines.

$j = 1, \ldots, D$, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, the associated robot joint configuration $\mathbf{y}^* = [y_1^*, \ldots, y_D^*]^T$ can be estimated as

$$y_j^* = \mathbf{K}_j(\mathbf{x}^*, \mathbf{X})[\mathbf{K}_j(\mathbf{X}, \mathbf{X}) + \sigma_{j,\mathrm{n}}^2 \mathbf{I}]^{-1} \mathbf{y}^j. \quad (9)$$

The coefficients of matrix $\mathbf{K}_j$ are defined as

$$(\mathbf{K}_j(\mathbf{X}', \mathbf{X}''))_{k,l} = \mathrm{k}_j(\mathbf{x}'_k, \mathbf{x}''_l), \quad (10)$$

where k_j is the selected real kernel function (see below) and $\mathbf{X}' = [\mathbf{x}'_1, \ldots \mathbf{x}'_{N'}]$, $\mathbf{X}'' = [\mathbf{x}''_1, \ldots \mathbf{x}''_{N''}]$. Thus $\mathbf{K}_j(\mathbf{x}^*, \mathbf{X}) \in \mathbb{R}^{1 \times N}$ and $\mathbf{K}_j(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{N \times N}$. The variance of the predicted values can be estimated as

$$\begin{aligned} \mathrm{cov}(y_j^*) = \; & \mathbf{K}_j(\mathbf{x}^*, \mathbf{x}^*) - \\ & \mathbf{K}_j(\mathbf{x}^*, \mathbf{X})[\mathbf{K}_j(\mathbf{X}, \mathbf{X}) + \sigma_{j,\mathrm{n}}^2 \mathbf{I}]^{-1} \mathbf{K}_j(\mathbf{X}, \mathbf{x}^*). \end{aligned}$$

One commonly used kernel function is

$$\mathrm{k}_j(\mathbf{x}', \mathbf{x}'') = \sigma_{j,\mathrm{f}}^2 \sum_{i=1}^{3} \exp\left(-\frac{1}{2} \frac{(x_i - x'_i)^2}{l_{j,i}^2} \right), \quad (11)$$

which results in a Bayesian regression model with an infinite number of basis functions. The parameters $\{\sigma_{j,\mathrm{f}}, \sigma_{j,\mathrm{n}}, l_{j,1}, l_{j,2}, l_{j,3}\}_{j=1}^{D}$ are called hyperparameters and need to be estimated by an off-line nonlinear optimization process. See [18] for more details.

An active eye system is crucial for reliable object fixation with a humanoid robot. On the other hand, error in the estimated positions increases when the eyes are active [22]. Fig. 4 shows that correction by Gaussian process regression can successfully cancel out a part of the estimation error, which enables the robot to touch an object even though the estimated 3-D object positions are fairly inaccurate.

To generate a pushing movement, we first estimate the center point of all 3-D features included in the initial object hypothesis. A probing push can be started from a position sufficiently displaced from this central position. This displacement is generated along a vector parallel to the table with a randomly selected direction. The end position is chosen to be on the other side of the object along the selected pushing vector through the center point.

The simplest way to compute a pushing movement is to generate a straight line between the two end-points and to move the robot hand along this straight line parametrized by time, using function (8) instead of the standard forward kinematics. However, a time-parametrized movement along the straight line is not always suitable for movements in unstructured environments, which are often perturbed and need to be adapted with respect to sensory signals. We therefore decided to generate pushing movements using a discrete pattern generator based on autonomous dynamic systems. The application of dynamic systems as policy primitives is closely related to the idea of motor pattern generators in neurobiology [19]. While general discrete arm movements require the introduction of a nonlinear component like for example the one introduced in [9], this was not necessary for the generation of probing pushing movements. We employed the following linear system to generate the desired point-to-point movements

$$\tau \dot{r} = \alpha_g (g - r) \quad (12)$$

$$\tau \dot{z} = \alpha_z (\beta_z (r - y) - z), \quad (13)$$

$$\tau \dot{y} = z. \quad (14)$$

Here y is one of the degrees of freedom that define the robot configuration $\mathbf{y}$ from Eq. (8), and z and r are auxiliary variables. It is easy to show that the above system is critically damped and that it has a unique attractor point at g for $\alpha_z = 4\beta_z > 0, \alpha_g > 0, \tau > 0$. System (12) – (14) is suitable for the generation of probing pushes because it is guaranteed to converge to g in a smooth manner regardless of the starting position and perturbations. In addition, the speed of movement can be modulated with parameter τ and even if the end configuration g is changed on the fly, the movement remains smooth up to the second order.

We generate a probing pushing behavior by executing a sequence of five dynamic systems (12) – (14), which result in the following movements

- Relocate the hand from its initial position to the position above the starting point for the pushing movement (leftmost image in Fig. 5).
- Move the hand towards the initial position for pushing (second image left in Fig. 5).
- Move the hand from the initial to the end position calculated as described above, thus generating the probing push (from second to fourth image in Fig. 5).
- Move the hand to a position above the end position for pushing (rightmost image in Fig. 5).
- Move the arm away from the viewfield of the robot.

The resulting probing movements are also shown in the video that accompanies this paper. With such a sequence of movements we reduce the possibility that the robot bumps into entities that are not included in the initial object hypothesis,

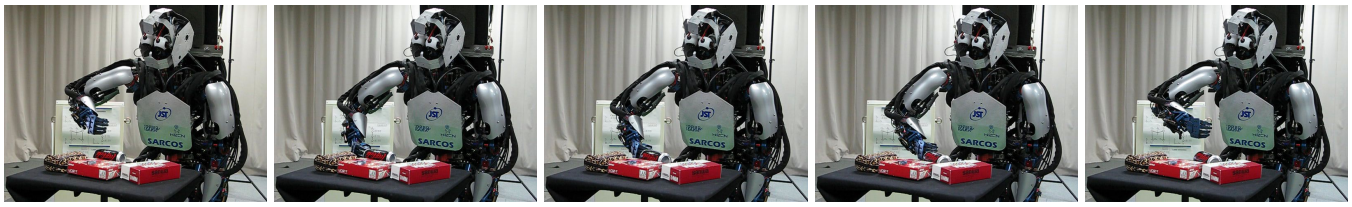


Fig. 5. A successful probing push. The robot starts at the position above the object, moves to the starting position for pushing, applies the probing pushing movement, and withdraws to the position above the object. All movements are generated using linear, autonomous dynamic systems. After the push the robot removes the arm from the viewfield to allow for unobstructed acquisition of the object image.

which reduces the danger of damaging the robot. Note that to generate such movements we need to train two functions (8); one to convert 3-D positions above the table into robot configurations and the second to convert 3-D positions on the table into robot configurations. In our experiments we acquired such data by kinesthetic guiding, where the robot arm was lead to a number of positions on and above the table, simultaneously estimating the resulting hand positions by active vision and saving the associated joints as sensed by proprioception.

In theory, a Cartesian straight line movement is more appropriate for probing pushes than a movement generated by a discrete dynamic system. However, since unknown objects cannot be located precisely and because of vision errors, it is not surprising that we observed no performance differences in our object learning experiments when we compared the proposed system with the pushing movements along straight-lines in Cartesian space. Note also that it is possible to utilize dynamic systems to generate Cartesian straight line movements by introducing a nonlinear component into system (12) – (14), like for example in the dynamic movement primitives methodology proposed in [9]. The advantage of doing this compared to straightforward time parametrization is that nonlinear dynamic movement primitives retain all positive properties of system (12) – (14) with respect to the movement modulation and robustness against perturbations. As explained above, it was not necessary to follow this route in our experiments.

A. *Hypothesis Validation*

After an object has been pushed, the Harris interest points and color MSERs have to be detected in the new camera images to verify if one of the hypotheses has moved. To match the interest points, we use the SIFT descriptor [14], which has proven to be descriptive and robust to small transformations. For the color MSERs, we use a rating calculated from the ratio of the length of the two principal axes of the region, and the average hue and saturation of the pixels belonging to it.

As the SIFT descriptors are sensitive to changes in scale and rotations in depth, we associate several descriptors with each point. After a push, we add descriptors at three different scales to the points that have been confirmed. When the number of descriptors associated with a point grows above a certain limit, we apply a k-means clustering to reduce it to the half of that limit. In this way the points can be tracked with high reliability, especially when descriptors from different viewing angles have already been accumulated.

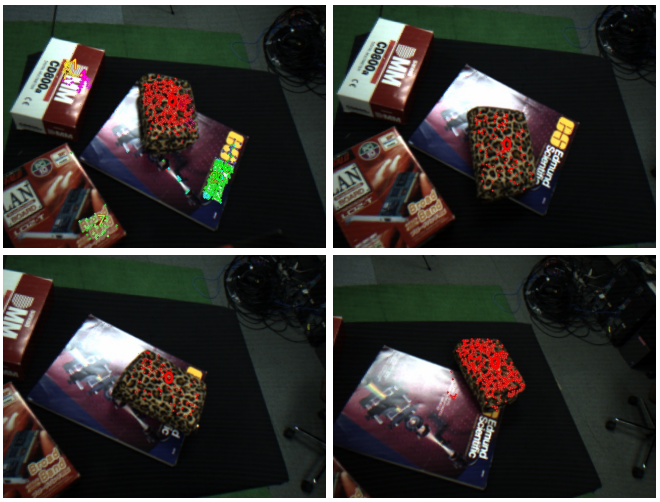


Fig. 6. The extracted features as seen from the robot eyes. The upper left image shows the initial object hypotheses. Hypothesis 0, which contains the largest number of feature points, was selected to generate the initial push. The upper right image shows the confirmed object feature points after the first push. The robot then continues pushing the object to acquire more object snapshots from different viewpoints. Note that the head and eyes are active to ensure that the object remains within the robot's viewfield. The number of extracted features can vary considerably from snapshot to snapshot. The acquired feature points are used to train a bag-of-features classifier.

IV. Object Learning and Recognition

The validated object hypothesis can be extended in the course of several push-and-verification steps, by adding new feature points that move consistently with the object or lie within its extent, and are verified or discarded after each push. As the object becomes visible from different directions, its visual appearance can be learned from multiple viewpoints. Features get out of sight when the object is rotated, in which case they are either simply not found or they are mismatched to different feature points. To prevent problems that would arise from mismatched features, a validated feature point that does not move in unison with the hypothesis is not used for the estimation of motion at the next step, and if it does so twice, it is completely discarded.

To encode the visual appearance of an object, we create a bag-of-features model (BoF) as introduced in [2], which is a histogram of the occurrences of feature descriptors

that are assigned to clusters learned from a large number of training features. We create the BoF model using SIFT descriptors of the verified feature points belonging to the object hypothesis. To include color information, we do not directly use color MSERs, but instead create a saturation-weighted hue histogram [23] within the ellipse spanned by the principal axes of the set of confirmed interest points and MSER centers. The BoF model and the hue histogram together form an object descriptor that incorporates both local greyscale descriptors of salient points and global color information.

After each push and subsequent validation of the points and MSERs belonging to the hypothesis, two object descriptors are saved. One is created using all validated features that have been accumulated so far, with the intent to obtain a comprehensive description of the object. The other uses only those validated features which are visible at that instant, thus having a snapshot-like character. Depending on the number of pushes, several descriptors are created and saved for each object that needs to be learned.

For object recognition, the descriptor of the considered hypothesis is calculated and compared to the stored descriptors of known objects. As a distance measure between the two descriptors, we use the weighted sum of normalized χ^2 histogram distances of the BoF model and the hue histogram. Both histogram distances are normalized individually by dividing them by the average distance of the hypothesis to all stored histograms. For recognition, we then apply a k-nearest-neighbors decision.

The performance of bag-of-features based recognition strongly depends on the successful segmentation of the object that needs to be recognized. The segmentation problem is often resolved by statistical feature clustering and by regular or randomized windowing [17]. As the segmentation problem is identical to the one that we face during the learning process, we use our hypothesis generation and active segmentation approach also to support recognition. By pushing the object several times, we achieve very high recognition rates due to the highly accurate segmentation.

A. Experimental Results

Since pushing induces a rather uncontrolled object motion, it is of crucial importance for the success of the learning process that the robot does not loose track of the object. The SIFT descriptor is sensitive to large changes in scale and rotations in depth, therefore large translations in the direction of the camera axis or significant rotations may be harmful, while a translation in the image plane causes no problems. Table I shows with which reliability the object is recovered after a motion along the camera axis. Enlarging the distance from the camera by a certain factor causes a scale change of the same factor. As can be seen, moving the object over a distance of up to 30% of its distance to the camera is unproblematic, above that value there is an increasing risk of loosing track. In practice this means that even for a rather small object-camera distance of 50 cm, a translation of 15 cm is safe.

TABLE I

OBJECT RECOVERY RATE AFTER MOTION IN DEPTH

distance ratio	1.2x	1.3x	1.4x	1.5x	1.6x
recovery rate	100 %	100 %	91 %	54 %	4 %

TABLE II

OBJECT RECOVERY RATE AFTER ROTATION

rotation angle	20°	30°	40°	50°	60°
recovery rate	100 %	100 %	83 %	56 %	11 %

TABLE III

OBJECT RECOGNITION RATE FOR THE INITIAL HYPOTHESES AND AFTER A FEW PUSHES

init. hyp.	1 push	2 pushes	3 pushes
77 %	86 %	96 %	98 %

Greater peril arises from rotations in depth. Table II shows the sensitivity of our approach to such transformations. While a change in orientation of the object of up to 30° is not a big problem, larger rotations may lead to the object not being recovered after the push. Therefore, if the pushing strategy is designed with the intent to reveal different sides of the object, it is safer to execute many small rotations instead of a few large ones.

To test the usefulness of the obtained object representation for recognition, we learned the appearance of 25 objects from different viewing directions (20 histograms for each object). As recognition is based on a bag-of features model and on a global hue histogram of the object, it is necessary to first segment the object. Then the BoF and hue histogram are calculated, and a 3-nearest neighbors decision based on the χ^2 histogram distance to the known objects is made.

To evaluate the performance of the recognition system, we tested using our initial hypothesis generation (see Sec. II) as well as the validated hypotheses after the probing pushes. Table III shows the recognition accuracy for the initial hypotheses and for confirmed hypotheses after $1-3$ pushes. As can be seen, a combination of the greyscale-based BoF and hue histogram allows for very reliable recognition. In the process of iterative pushing and verification, false features are discarded and an increasingly complete object representation is obtained, which leads to nearly error-free recognition after a few pushes.

V. CONCLUSION

While in this paper we focused on autonomous acquisition of object models, our system allows the accumulation of knowledge from different sources. Models can be acquired either from large databases of stored models, in interaction with a human teacher where the human teacher performs the pushes instead of the robot, or fully autonomously. Such an approach is essential to prevent on the one hand excessively long learning times and on the other hand to enable acquisition of new knowledge as need arises. We

believe that our integrated approach makes an important step towards truly autonomous robots.

VI. Acknowledgments

Research leading to these results was supported in part by the EU Seventh Framework Programme under grant agreement no. 270273, Xperience, and by "Brain Machine Interface Development", SBRPS, MEXT. A. Ude would like to thank NICT for its support within the JAPAN TRUST International Research Cooperation Program.

References

[1] T. Chaperon and F. Goulette. Extracting cylinders in full 3d data using a random sampling method and the gaussian image. In *Proc. Vision Modeling and Visualization Conference*, 2001.

[2] G. Csurka, C. Dance, L. X. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. ECCV Int. Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, 2004.

[3] J. Feldman. What is a visual object? *Trends in Cognitive Sciences*, 7(6):252–256, 2003.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, volume 24, 1981.

[5] P. Fitzpatrick. First contact: an active vision approach to segmentation. In *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 2161–2166, Las Vegas, Nevada, 2003.

[6] P. Forssen. Maximally stable colour regions for recognition and matching. In *IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[7] J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, MA, 1979.

[8] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, page 147151, 1988.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398–1403, Washington, DC, 2002.

[10] D. Kraft, N. Pugeault, E. Baseski, M. Popovic, D. Kragic, S. Kalkan, F. Wörgötter, and N. Krüger. Birth of the object: Detection of objectness and extraction of object shape through object-action complexes. *Int. J. Humanoid Robot.*, 5(2):247–265, 2008.

[11] M. Krainin, P. Henry, X. Ren, and D. Fox. Manipulator and object tracking for in-hand 3D object modeling. *Int. J. Robotics Res.*, 2011 (online first).

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2169–2178, New York, NY, 2006.

[13] W. H. Li and L. Kleeman. Segmentation and modeling of visually symmetric objects by robot actions. *Int. J. Robotics Res.*, 30(9):1124–1142, 2011.

[14] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. Int. Conf. Computer Vision*, Corfu, Greece, 1999.

[15] J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conference*, 2002.

[16] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proc. 17th Int. Conf. Machine Learning*, San Francisco, CA, 2000.

[17] A. Ramisa, S. Vasudevan, D. Scaramuzza, R. L. de Mántaras, and R. Siegwart. A tale of two object recognition methods for mobile robots. In *Proc. 6th Int. Conf. Computer Vision Systems*, 2008.

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

[19] S. Schaal and D. Sternad. Programmable pattern generators. In *Proc. Int. Conf. on Computational Intelligence in Neuroscience*, Research Triangle Park, NC, 1998.

[20] E. Stergaršek-Kuzmič and A. Ude. Object segmentation and learning through feature grouping and manipulation. In *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*, pages 371–378, Nashville, TN, 2010.

[21] A. Ude, D. Omrčen, and G. Cheng. Making object learning and recognition an active process. *Int. J. Humanoid Robot.*, 5(2):247–265, 2008.

[22] A. Ude and E. Oztop. Active 3-D vision on a humanoid head. In *Proc. 14th Int. Conf. Advanced Robotics*, Munich, Germany, 2009.

[23] K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 1582–1596, 2010.

[24] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *Int. J. Comput. Vision*, 73(2):123–138, 2007.