Exploration and Planning in a Three-Level Cognitive Architecture

D. Kraft, E. Başeski, M. Popović,
A. M. Batog, A. Kjær-Nielsen and N. Krüger
University of Southern Denmark, Denmark
{kraft,emre,akn,norbert}@mmmi.sdu.dk,
{mipop05,anbat06}@student.sdu.dk

R. Petrick, C. Geib, N. Pugeault and M. Steedman
University of Edinburgh, United Kingdom
{R.Petrick,C.Geib,npugeaul,steedman}@ed.ac.uk

T. Asfour and R. Dillmann
University of Karlsruhe (TH), Germany
{asfour,dillmann}@ira.uka.de

S. Kalkan and F. Wörgötter
University of Göttingen, Germany
{sinan, worgott}@bccn-goettingen.de

B. Hommel
Leiden University, The Netherlands
hommel@fsw.leidenuniv.nl

R. Detry and J. Piater
University of Liege, Belgium
{renaud.detry,justus.piater}@ULg.ac.be

***Abstract*— We describe an embodied cognitive system based on a three-level architecture that includes a sensorimotor layer, a mid-level layer that stores and reasons about object-action episodes, and a high-level symbolic planner that creates abstract action plans to be realised and possibly further specified by the lower levels. The system works in two modes, exploration and plan execution, that both make use of the same architecture. We give results of different sub-processes as well as their interaction. In particular, we describe the generation and execution of plans as well as a set of learning processes that take place independently of, or in parallel with, plan execution.**

I. Introduction

In this paper we describe a cognitive system consisting of three hierarchically nested layers: a low-level, sensorimotor layer connecting sensory processors to motor procedures; a mid-level layer that stores object-action episodes in the form of Object-Action Complexes (OACs) and reasons about memorised events; and a high-level symbolic planner that creates abstract action plans to be realised, and possibly further specified, by the lower levels. The cognitive system works in two modes. In the first mode, the system explores its environment and learns about object-action associations. In the second mode, it constructs and performs directed plans of action. In both modes it is able to recover from unexpected errors that arise through its interaction with dynamic environments, and to learn from such events to improve its future performance.

We introduce a software architecture and its application in a concrete embodied system consisting of an industrial 6 degrees of freedom (DoF) robot with a two finger grasper, haptic sensing by means of a force-torque sensor and a high resolution stereo system in which a preliminary attention system is realized (see Fig. 1). The system also includes a symbolic planner, and is equipped with a memory of past experiences that enables certain learning processes. The task we address in this paper is a simple table cleaning scenario, however, our proposed architecture allows for more general tasks.

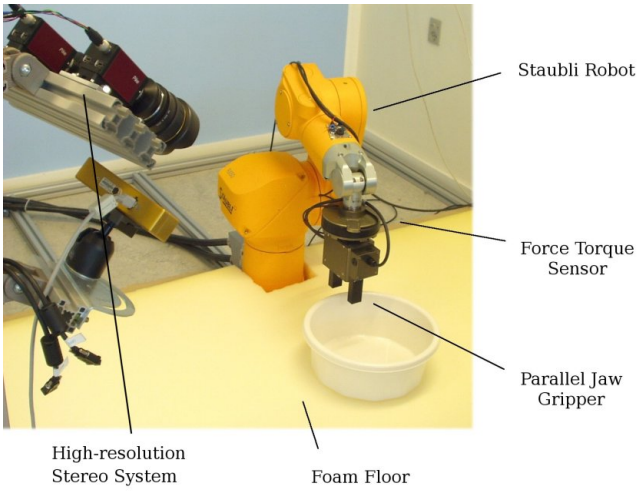


Fig. 1. Embodiment of the Cognitive System

The aim of this architecture is not only to solve certain prescribed tasks with the best possible performance, but also to define a set of cognitively rich processes in which the robot is able to (i) construct a plan to solve the given task, (ii) execute the plan in a dynamic environment, and (iii) recognise and react to unexpected events that arise during exploration and plan execution. For (iii), in particular, the robot can address such issues at either the robot-vision level (e.g., by withdrawing from collisions, see Fig. 2), the mid-level (e.g., by reinspecting the scene at a higher resolution to gain additional information about the worlde, see Fig. 3), or at the planning level (e.g., by complete replanning). During all such interactions with its environment, the robot represents its experiences in terms of Object Action Complexes (OACs) [1] that become stored and utilised by various learning processes

in the system.

In its early stages, this work focuses on a very limited domain: objects become represented as 3D circles to which grasps become associated (see Fig. 4). This limitation is merely for development purposes and work is ongoing to extend the proposed system to arbitrary objects (see [2] and [3]) The only fundamental restriction of our proposed architecture stems from limitations in the vision system (namely, pose estimation of arbitrary objects [3]) and grasping modules (the grasping device only allows up to grasp a limited range of objects). In this paper we report on both our current progess and proposed extension to our limited domain.

(a) (b) (c)

Fig. 2. Withdrawal after the detection of a collision. (a) Approaching the object, (b) a collision is detected, (c) start of withdrawal action.

The paper is structured as follows: section II introduces the system architecture, section III describes the system's embodiment and visual representation, section IV describes the mid-level, and section V describes the planning level. The relationship between our architecture and other cognitive architectures is discussed in detail in section VI.

II. ARCHITECTURE

The architecture of our system consists of three levels: a sensorimotor robot-vision level providing multi-sensorial information and action options, a mid-level component at which the robot's past experiences in the world are stored and made available to various learning processes in the architecture, and a symbolic planning system that generates plans based on the information provided by the robot-vision and mid-level components, and which also monitors the execution of these plans. This architecture integrates several diverse approaches from computer science, artificial intelligence, and cognitive psychology, and addresses a variety of learning and adaptation problems at all three levels.

The robot-vision level provides visual and tactile information to the higher levels and manages robot-level action commands. For instance, the robot-vision level is able to perform operations such as "grasp object B using grasp type A", start an explorative action such as "poking", or shift its visual attention to a certain location in the scene. Moreover, it has (pre-programmed) control mechanisms that allow it to detect unexpected events, and avoid certain consequences that would normally lead to emergency stops and/or damage objects or the robot itself (see Fig. 2).

In its current (preliminary) state, the mid-level is responsible for storing OACs in memory, controlling access to this information by different learning processes, and for refining grasping reflexes and object-action models based on the stored OACs. It is also responsible for transforming

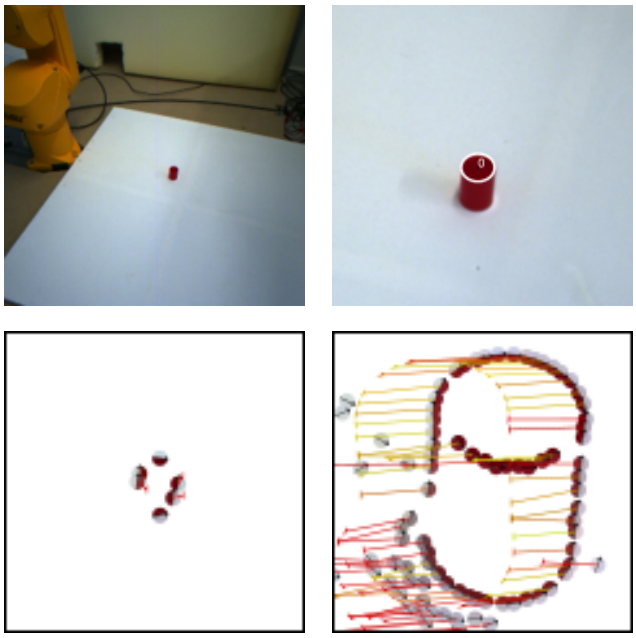
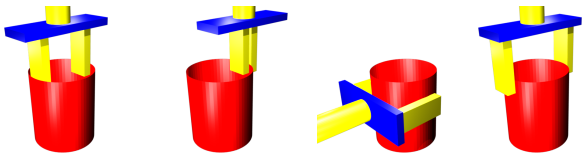


Fig. 3. Circle detection is not successful (left) because of the small number of feature descriptors extracted from a downsampled version of the high resolution images. It is successful (right) when the system focuses on the object at full resolution.



A B C D

Fig. 4. Grasp types available in the example domain. They are defined object centric and based on the upper circle.

transient sensorial input into messages that are passed on to the planning level.

The planning level is responsible for constructing high-level, goal-oriented plans and for feeding these plans to the lower system levels for execution by the robot. To do so, the planning system maintains an abstract model of the objects, properties, and actions available to the robot in the world. The planner also receives regular updates on the state of the world from the lower levels, which it uses to monitor the success of plans being executed, to control resensing and replanning activities in the system, and to update its internal models.

III. EMBODIMENT AND VISUAL REPRESENTATION

This section presents the robot-vision level of the architecture. The system's embodiment is described in section III-A. The visual representation used for object localisation, learning, and the generation of grasping affordances is briefly described in section III-B. Grasp part associations used in planning and exploration are described in section III-C.

A. Embodiment

To enable interactions with the real world, we use a robot-vision system with the components shown in Fig. 1. The components are:

- The workspace is observed using a 2024×2024 pixels high resolution camera system. In normal working mode downsampled images of the whole scene are used, but it is also possible to get a full resolution image of a region of interest. The camera system is calibrated relatively to the robot.
- A 6-DoF industrial robot is used, together with a two fingers grasper. This enables the system to grasp objects at different locations in the scene.
- A 6-DoF force-torque sensor mounted between robot and grasper allows for the measurement of forces at the wrist. It is used to detect collisions and to back off when a collision is detected (see Fig. 2). To limit the build up of high forces during the reaction time of the system a foam layer was placed on the floor of the scene.

B. Representation of visual information

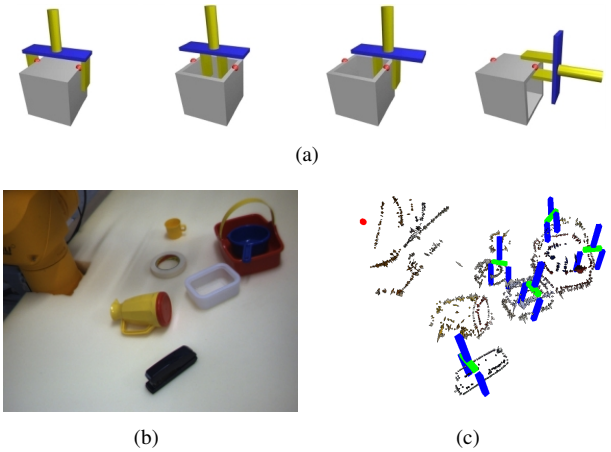
This work uses the hierarchical representation presented in [4]. An example is presented in Figure 5, which shows what kind of information is processed on the different representation levels. At the lowest level of the hierarchy is the image's pixel RGB values (Fig. 5(a)). The second level processes the results of local filtering operations (Fig. 5(b)), that give rise to the multi-modal 2D primitives at the third level (Fig. 5(c)). This third level processes not only the 2D primitives, but also 2D contours (Fig. 5(d)) created using perceptual organisation (see [5]). The last level contains 3D primitives and 3D contours (Fig. 5(e-f)) created using stereopsis on a pair of the previous level's representations.

C. Grasp-Part Association

Objects that share common features (parts) often afford the same actions (e.g., an object with an handle can be picked up by grasping it through the handle). We exploit these common parts to initiate learning by transferring the previous experience with one object to another.

In [6], coplanar line-segments are used to predict different grasp types — Fig. 6 shows more recent results on this approach. This can be either seen as a complex reflex — used for generating ground truth for grasp learning — or as a basic form of part-action association. The circle-grasp relation used in the scenario described herein uses a more complex part (the circle). Currently, the associated grasps/part associations are predefined. However, we aim at a learning of these associations from experience. A first step in this direction is described in section IV-A.

Note that our grasping stretagy is very much based on 3D contour information which we find complementary to other approaches (see, e.g., [7] that makes use of 3D shape information). For a discussion of underlying neuronal mechanisms in the context of grasping we refer the reader to, e.g., [8].



(a) (b) (c)

Fig. 6. Coplanar primitives can predict a grasp (see [6]). (a) Different grasp types based on coplanar primitives (marked as red dots). (b) Example scene. (c) The (automaticly selected) best five grasping hypotheses in the example scene.

IV. Mid-level

The mid-level is responsible for a number of tasks (from which some are not yet or only rather naively implemented). In our system, on the mid-level the storage of information used for additional learning processes (as well as the learning as such) is organised (section IV-A and section IV-B). Also the temporal consistency of the permanently varying information on the raw signal level which is required by the planning level is provided (section IV-C). It is partly motivated by the idea that perceptual events and actions are cognitively represented and integrated in a common store [9]. Apart from organising the storage of information (sections IV-A and IV-B) the mid-level also provides episodic pointers to perceptually available objects (i.e., a kind of working memory: section IV-C).

A. Refinement of Grasping Strategy

The grasping affordances associated to the parts (see section III-C) is initially hardwired and, in case of success, give rise to an additional object learning process that requires physical control over the object as achieved after a successful grasp (this is described in section IV-B). Moreover, the grasping behaviour linked to these part affordances generates labelled data that can be used for further learning processes: Since for each attempted grasp success or failure can be measured by the distance between the hand's two fingers after a lifting operation, a large number of training data becomes generated.

In case of the "circle-reflex", it is a priori unclear which grasp type can be used for which circle radius. Fig. 7(d) shows distribution of successes and failures for grasp type A generated over a large number of trials. The distributions in real and artificial scenarios are rather different (as shown in Fig. 7(a)). This is because grasps in a real scenario might fail, due to external reasons such as collision before the grasp, or imprecision in the 3D circle reconstruction. Also, grasps might be successful by accident, e.g., when a planned grasp

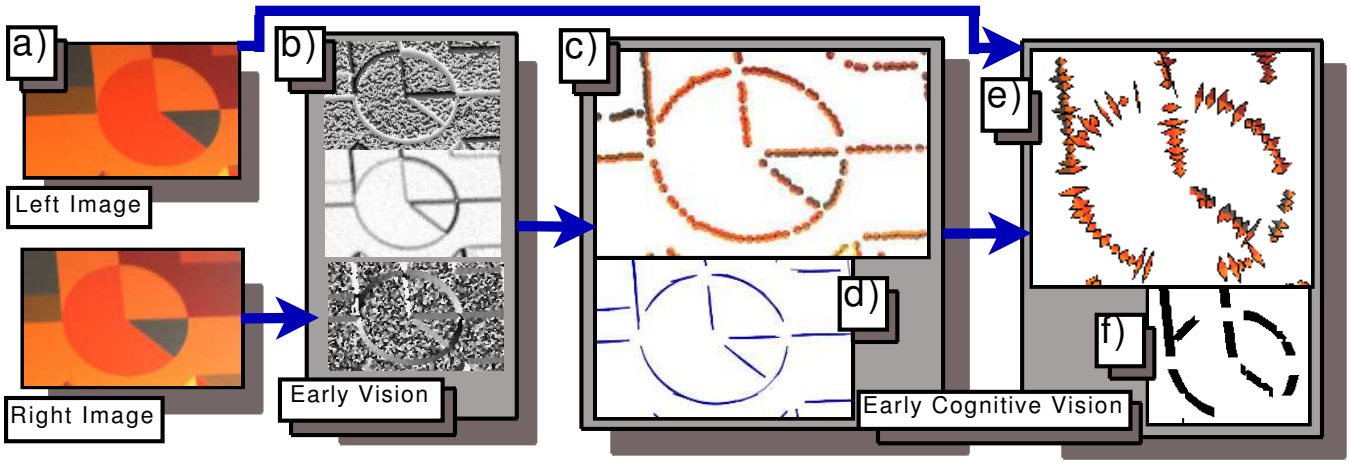


Fig. 5. The different types of visual information processed by the hierarchy of representations **(a)** Original image, **(b)** Filtering results, **(c)** 2D primitives, **(d)** 2D contours, **(e)** 3D primitives, **(f)** 3D contours.

of type A is in reality of another type. Nevertheless, the experiences give valuable information and can be translated into likelihoods of success depending on the radius (see Fig. 7(f)) that can then be used by higher level processes to decide what grasp type should be selected. Note that learning can also be applied in the more difficult case of the coplanarity-based grasping reflex, since success can be measured in the very same way. This, however, requires a much more complex learning framework (see [3]).

B. Birth of the Object

In addition to the refinement of pre-wired reflex behaviours, the exploration behaviour allows the proposed system to learn a three-dimensional visual representation of shape. In [2] we have presented an algorithm that, based on the initial grasping reflexes and on the knowledge of the robot's arm motion, learn models of 3D objects' shape.

A successful grasp on an object endows the robot with control over this object. The system will then manipulate the object to visually inspect it from a variety of viewpoints. The object's motion during this inspection phase is known to be the same as the robot's arm's. This knowledge of the object's motion allows to infer predictions on the visual representation extracted at later stages. Tracking objects visual representations allows to improve the internal representation of their shape, in three respects:

- Because only the object moves as predicted by the robot's arm motion, tracking of visual primitives allows to segment the object from the rest of the scene;
- The integration in the same coordinate system of visual information gathered from multiple viewpoints allows to generate a representation of the object's full 3D shape;
- Tracking aspect's of the object representation allows to reduce inaccuracy in the shape representation.

The result of this process is a full 3D representation of the object shape, with an associated knowledge of the grasp that was successful.

Fig. 8 shows the results of object learning. Once the object's shape is known to a satisfying level, the planner

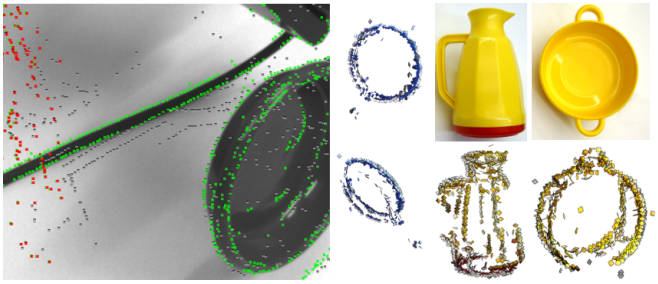


Fig. 8. Birth of the Object. On the left hand side, the dots on the image shows the predicted structures. Spurious primitives and parts of the background are not confirmed by the image, and are shown in grey. The confirmed predictions are shown in green. The middle shows the shape model learned from this object. The right hand side shows the shape model learned from tow other objects. Note that the gap in the shape models correspond to where the robot's hand held the objects.

(see section V) is informed of the new object discovery and of the associated grasp.

Current work endeavour to use the acquired representation for object recognition and pose estimation, therefore extending the simplified 'circle scenario' into a more general object context.

C. Temporal Consistency

The information provided by the robot-vision layer is intrinsically noisy. This leads to

- phantom objects appearing (i.e., false positives),
- objects present in the scene not being detected,
- object labels not being consistent over time.

While the first two cases are not frequent, the third case is a constant problem. The planning layer needs accurate state information, therefore these problems need to be corrected before the state information is sent to the planner.

These problems become solved by matching the last know state with the sensed information. Objects that are detected in the sensory data that are close in position, orientation, and radius to one object in the last known state are assumed to be the same. This solves the object labelling problem. When

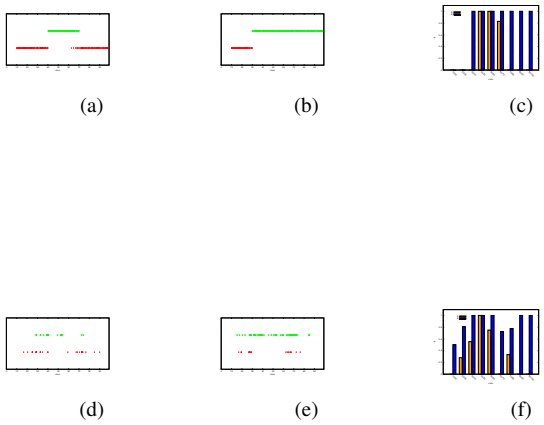


Fig. 7. Grasp experiences and success distributions. (a)-(c) show the results for artificial data, while (d)-(f) show data from real experiences. (a) and (d) show grasping experiences made for grasp type A for different radiuses while (b) and (e) show the same for grasp type B. In these diagrams the green crosses (in the upper row) represent a successfull grasp, while the red crosses (lower row) represent failures. (c) and (f) show success probabilities for the two grasp in discrete radius bins. These can be used as a indicator which grasp to choose for a unknown grasping situation.

no object in the old scene can be matched to a new object, the new object is considered wrong. Undetected objects are treated in the same way. This part of the system is still in a very premature mode, and future work has to address complex issues such as object permanency [10] under occlusions or accidental displacements of objects under unplanned contacts of objects and robot.

V. Planning

The high-level planning component of the system is responsible for constructing action plans that direct the behaviour of the robot in order to achieve a specified set of goals. (For instance, in our example scenario a plan might be constructed to clear all the open objects from the table.) The planning system consists of three main parts: the *high-level domain model*, the *planner* itself (in this case, the PKS planner [11], [12]), and the *plan execution monitor*. In particular, we utilize a state of the art planning system, and are motivated by *learning* high-level domain models through the robot's interaction with the environment. Together, these components are responsible for the high-level representation and reasoning in the system, and are connected to other systems through a communication architecture which controls the flow of information between the system components.

A. High-level domain model

The high-level domain model consists of a formal representation of the robot's world, described in a STRIPS-like [13] language used as input to the planner. In particular, the domain model specifies the objects and properties in the world, and the actions available to the planner for directing the robot. Currently, some parts of the domain model are automatically induced through the robot's experiences in the world, and we are investigating how other aspects of this model can be learned through machine learning techniques.

1) Objects: Objects are simply labels (strings) that denote actual objects in the real world. (E.g., *obj1* may represent a particular red block on the table in the robot's workspace.) Object names do not typically change over time and always refer to the same world-level objects. As a result, planning-level objects also act as indices to actual real-world object information stored at the robot and memory levels, and can be used by all system levels to refer to an object uniquely.

2) Properties: Properties are specified by predicates and functions that denote particular qualities of the world, robot, and objects. High-level properties are typically quite abstract and correspond to combinations of concepts available at the lower levels. For instance, we define the following properties in our example scenario:

- *open*(x) - object x is open,
- *gripperempty* - the robot's gripper is empty,
- *ingripper*(x) - the robot is holding x in its gripper,
- *ontable*(x) - object x is on the table,
- *onshelf*(x) - object x is on the shelf,
- *isin*(x, y) - object x is stacked in object y,
- *clear*(x) - no object is stacked in object x,
- *instack*(x, y) - object x is in a stack with y at its base,
- *radius*$(x) = y$ - the radius of object x is y,
- *shelfspace* $= x$ - there are x empty shelf spaces.

In this case, *gripperempty* closely corresponds to a sensor that detects whether the gripper can be closed without contact, while *ontable* requires a conjunction of data from the visual sensors concerning object positions. Parametrized properties can also be instantiated by specific domain objects. Thus, *ontable*$(obj1)$ means "object *obj1* is on the table" and *ingripper*$(obj2)$ means "object *obj2* is in the gripper."

3) Actions: Actions represent high-level counterparts to some of the motor programs available at the robot level. Unlike low-level motor programs, high-level actions are modelled with a high degree of abstraction that incorporates state-specific elements into their operation. For instance, the following actions are defined in our example domain:

- *graspA-table*(x) - grasp x from the table using grasp A,
- *graspA-stack*(x) - grasp x from a stack using grasp A,
- *graspB-table*(x) - grasp x from the table using grasp B,
- *graspC-table*(x) - grasp x from the table using grasp C,
- *graspD-table*(x) - grasp x from the table using grasp D,
- *putInto-object*(x, y) - put x into an object y on the table,
- *putInto-stack*(x, y) - put x into y at the top of a stack,
- *putAway*(x) - put object x away on the shelf,
- *sense-open*(x) - determine whether x is open or not.

Our high-level actions do not require 3D coordinates, joint angles, or similar real-valued parameters. Instead, they are defined in an *object-centric* manner, with parameters that can be instantiated with specific objects. Two types of actions are described: *physical actions* that change the state of the world, and *sensing actions* that observe the state of the world.

The physical actions in our example domain include actions for manipulating objects in the world. The first five actions indicate the possible grasping options and correspond to the four types of grasps available to the robot (see Fig. 4). Grasp A is divided into two separate actions that account for different object configurations (i.e., an object on the table versus an object at the top of a stack). This avoids the need for conditional action effects in our representation. The two *putInto* actions similarly model different object destinations. The *putAway* action is a generic operation for moving a grasped object to a location on the shelf.

We also define a high-level sensing action, *sense-open*. This action models an operation that provides the planner with specific information about an object's "openness", without intentionally changing the object's state. At the robot/vision level, this action will ultimately be executed as either a physical test (e.g., poking an object to check its concavity) or a visual test (e.g., focusing on the object at a higher resolution). The mid-level memory is responsible for refining *sense-open* actions into robot/vision operations that are appropriate for the given object and context.

4) Learning: One of the strengths of our approach is that it enables us to *learn* certain aspects of the high-level domain. For instance, the set of domain objects is not fixed a priori but is induced through the robot's birth-of-an-object process: as objects are discovered in the world, the planner is informed as to their existence and new object labels are assigned. Newly learned objects are incorporated into the domain model to be used in future plan construction. We are also investigating machine learning techniques for learning high-level action descriptions. In particular, a kernel perceptron learning method is being tested to learn action effects from snapshots of world states made up of high-level properties. Preliminary results indicate efficient training with low average error rates ($<3\%$) when tested on the above example domain [14]. Although the properties in our domain are currently fixed, we are also exploring additional learning processes whereby new properties can be introduced into the domain model with the help of the middle level.

TABLE I
EXAMPLES OF PKS ACTIONS IN THE TABLE CLEARING TASK

Action	**Preconditions**	**Effects**
graspA-table(x)	K(*clear*(x)) K(*gripperempty*) K(*ontable*(x)) K(*radius*$(x) \geq$ *minA*) K(*radius*$(x) \leq$ *maxA*)	*add*$(K_f$, *ingripper*$(x))$ *add*$(K_f, \neg$*gripperempty*$)$ *add*$(K_f, \neg$*ontable*$(x))$
sense-open(x)	$\neg K_w$(*open*(x)) K(*ontable*(x))	*add*$(K_w$, *open*$(x))$

B. PKS planner

High-level plans are built using PKS ("Planning with Knowledge and Sensing") [11], [12], a state of the art planner that can operate with incomplete information and sensing actions. Unlike traditional approaches to AI planning, PKS operates at a higher "knowledge level" of abstraction, by modelling an agent's knowledge state. By doing so, PKS can reason efficiently about certain types of knowledge, and make effective use of non-propositional features, like functions, which often arise in real-world planning scenarios.

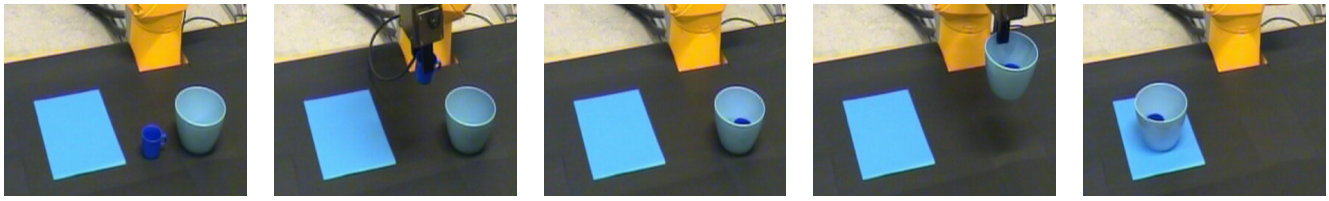
PKS is based on a generalization of STRIPS [13]. In STRIPS, a single database is used to represent the planner's complete world state. Actions update this database in a way that corresponds to their effects on the world. In contrast, PKS's representation goes beyond that of traditional STRIPS. In PKS, the planner's knowledge state is represented by five databases, each of which stores a particular type of knowledge. Actions are described in terms of the changes they make to the databases and, thus, to the planner's (typically incomplete) knowledge. Table I shows two PKS actions, *graspA-table* and *sense-open*. Here, K_f refers to a database that models knowledge of simple facts, while K_w is a specialized database that stores the results of sensing actions that return binary information.

Two different types of plans can be built in PKS: *linear plans* that are simply sequences of actions, and *conditional plans* that contain branches resulting from the inclusion of sensing actions. For instance, in the table cleaning task PKS can construct the simple plan:

$$\begin{array}{l} \textit{graspA-table}(\textit{obj1}), \textit{putInto-object}(\textit{obj1}, \textit{obj2}), \\ \textit{graspD-table}(\textit{obj2}), \textit{putAway}(\textit{obj2}), \end{array} \quad (1)$$

to put an object *obj1* into another object *obj2* before grasping the stack of objects and removing them to the shelf. In this case, the plan is linear since it only contains physical actions.

When a plan contains sensing actions, PKS can reason about the possible outcomes of such actions by adding conditional branches to the plan. For instance, if PKS is given the goal of removing the "open" objects from the table, but does not know whether an object *obj1* is open or not, then



(a) (b) (c) (d) (e)

Fig. 9. Performing of a Plan: (a) Initial scene. The small blue cup is *obj1*, while *obj2* stands for the light-blue bowl. The blue rectangle represents the shelf area. (b) Scene after executing *graspD-table*(*obj1*). (c) Scene after executing *putInto-object*(*obj1*, *obj2*). (d) Scene after executing *graspB-table*(*obj2*). (e) Scene after executing the final command *putAway*(*obj2*).

it can construct the conditional plan:

$$\begin{array}{l} \textit{sense-open}(\textit{obj1}), \\ \textit{branch}(\textit{open}(\textit{obj1})) \\ K^+ : \textit{graspA-table}(\textit{obj1}), \textit{putAway}(\textit{obj1}) \\ K^- : \textit{nil}. \end{array}$$

This plan first senses the truth value of the predicate *open*(*obj1*) and then branches on the two possible outcomes. When *open*(*obj1*) is true (the K^+ branch), *obj1* is grasped and put away; when *open*(*obj1*) is false (the K^- branch), no further action is taken.

C. Plan construction, execution, and monitoring

The planning level interacts with the rest of the system to construct and execute plans. For instance, the initial world state—forming the planner's initial knowledge state—is supplied to the planner from the robot/vision system. Once the planner has such information it constructs a high-level plan and feeds it to the robot, one action at a time, upon request from the lower levels. The planner can also send complete plan structures to the mid-level memory, to help it better direct the execution of robot-level actions. Upon action completion, the lower levels inform the planner as to any changes made to the world state, allowing the plan execution cycle to continue until the end of the plan. For instance, Fig. 9 shows the execution of the simple four step plan in (1). This plan does not contain any sensing actions and is constructed using objects in the initial scene.

A vital component in this architecture is the plan execution monitor, which assesses action failure and unexpected state information in order to control replanning and resensing activities. In particular, the difference between predicted and actual state information is used to decide between (i) continuing the execution of an existing plan, (ii) asking the vision system to resense a portion of a scene at a higher resolution in the hope of producing a more detailed state report, and (iii) replanning from the unexpected state.

The plan execution monitor also has the added task of managing the execution of plans with conditional branches, resulting from the inclusion of sensing actions, such as *sense-open*. When a sensing action is executed at the robot level, the results of the sensing will be passed to the planning level as part of a state update. Using this information, the plan execution monitor can then decide which branch of a plan it should follow, and feed the correct sequence of actions to the lower levels. If such information is unavailable, resensing or replanning is triggered as above.

VI. Discussion

We do not give a full description of other cognitive architectures (for this, see, e.g., [15], [16]), but instead highlight some important relations to prior work. Cognitive architectures often fall into one of two categories [16]. Some architectures are based on the notion of cognition as symbol manipulation, such as ACT-R [17] or SOAR [18], and focus on semantically-interpretable symbolic representations. They are often richly structured, contain a number of subsystems (e.g., workspaces and memory stores), and are very powerful in simulating higher-order cognitive processes, such as logical reasoning or constraint satisfaction. By emphasizing symbolic processing only, such architectures face serious grounding problems and are much better suited to model abstract thinking than sensorimotor control in changing environments [17]. Other architectures are based on the notion of distributed parallel processing, such as [19], [20], and focus on interactive subsymbolic representations, like neural networks. Such approaches do not tend to have dedicated subsystems or components, and are very powerful in simulating online, environmentally-sensitive behaviour. As a drawback, these approaches do not provide strong models of higher-level cognition, and their representations are often not well specified semantically [16].

Our approach aims at developing a hybrid cognitive architecture that takes the best of these two worlds, which are often taken as incompatible. We claim that high-level planning is not only easier to model using symbolic representations and rule-based operations, but we believe that a number of aspects of higher-order human cognition are more appropriately captured by the symbolic approach. At the same time, we consider a symbolic approach to low-level sensorimotor processing to be biologically implausible and too unreliable in a practical sense. Instead, we combine a subsymbolic online sensorimotor processing level with a symbolic level for abstract action planning and communication. Thus, both high-level planning and low-level exploration coexist within the same architecture. In this sense, our work is related to approaches like [21], but goes well beyond by allowing learning and adaptation at all levels of the architecture. Since planning and exploration is very much connected, we also extend on approaches that focus on learning and development

(as done, e.g., in [22]).

For instance, in terms of the question of symbolic versus sensory processing, we give one example of symbol grounding (see, e.g.,[23], [24]) in the context of the concept of "objects". The system described here (see section IV-B), is able to detect the category "objectness" as well as important physical properties by interacting with the world and by inducing predictable correlations in the sensory data (see also [25]).

Based on insights from cognitive neuroscience, we also suggest that a mid-level system is necessary to mediate between the low-level and high-level components. In humans, online visuomotor processing is mediated by the so-called dorsal pathway, which is evolutionarily old, cognitively rather inaccessible but fast and reliable [26]. In contrast, offline processing, such as higher-order perception and action selection, is carried out by the ventral pathway, which is evolutionarily younger, cognitively penetrable, highly interactive and informed by memory contents but rather slow [26]. Both pathways have particularly strong features and it is their interaction that allows for goal-directed and highly adaptive, yet fast and reliable, performance [27]. Obviously, what we consider the low processing level of PACO+ shares the main characteristics of the human dorsal pathway. At the same time, however, it is difficult to see how purely symbolic rule-based system can efficiently interact with online processing the same way as the human ventral system interacts with dorsal processing. Hence, we do not regard the characteristics of ventral processing to be well captured by our highest processing level but suggest a subsymbolic mid-level that mediates between on-line processing and symbolic high-level reasoning, and that interacts with the lowest level in a similar way as ventral and dorsal visuomotor pathways in humans.

VII. Conclusions

In this paper we introduced a three-level cognitive architecture within an embodied system that facilitates exploration, planning, and learning in dynamic environments. It also allows for extensions to be made to the application domain in which the robot-vision, mid-level, and high-level planner operate. While we are aware that some components of our proposed system are still at an early state of development, we also believe that the described interactions between the different levels and sub-modules shows great potential for our cognitive system.

Acknowledgement

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

[1] "Pacoplus: Perception, action and cognition through learning of object-action complexes," *IST-FP6-IP-027657, Integrated Project*, 2006-2010.

[2] N. Pugeault, E. Baseski, D. Kraft, F. Wörgötter, and N. Krüger, "Extraction of multi–modal object representations in a robot vision system," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2007.

[3] R. Detry and J. Piater, "Hierarchical integration of local 3d features for probabilistic pose recovery," *Robot Manipulation: Sensing and Adapting to the Real World, 2007 (Workshop at Robotics, Science and Systems)*, 2007.

[4] N. Krüger, M. Lappe, and F. Wörgötter, "Biologically motivated multi-modal processing of visual primitives," *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour*, vol. 1, no. 5, pp. 417–428, 2004.

[5] N. Pugeault, F. Wörgötter, and N. Krüger, "Multi-modal scene reconstruction using perceptual grouping constraints," in *Proc. IEEE Workshop on Perceptual Organization in Computer Vision (in conjunction with CVPR'06)*, 2006.

[6] D. Aarno, J. Sommerfeld, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft, and N. Krüger, "Early reactive grasping with second order 3d feature relations," *IEEE International Conference on Robotics and Automation (ICRA), Workshop: From features to actions - Unifying perspectives in computational and robot vision*, 2007.

[7] B. Bender and G. Bone, "Automated grasp planning and execution for real world objects using computer vision and tactile probing," *International Journal of Robotics and Automation*, vol. 19, no. 1, 2004.

[8] M. Goodale, J. P. Meenan, H. H. Blthoff, D. A. Nicolle, K. J. Murphy, and C. I. Raciot, "Separate neural pathways for the visual analysis of object shape in perception and prehension," *Current Biology*, vol. 4, pp. 604–610, 1994.

[9] B. Hommel, J. Müsseler, G. Aschersleben, and W. Prinz, "The theory of event coding (tec): A framework for perception and action planning," *Behavioral and Brain Sciences*, vol. 24, pp. 849–878, 2001.

[10] J. Piaget, *The psychology of intelligence*, 1976.

[11] R. P. A. Petrick and 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 (AIPS-2002)*. AAAI Press, 2002, pp. 212–221.

[12] ——, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*. AAAI Press, 2004, pp. 2–11.

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

[14] K. Mourão, R. P. A. Petrick, and M. Steedman, "Using kernel perceptrons to learn action effects for planning," in *International Conference on Cognitive Systems (CogSys 2008)*, 2008.

[15] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," Computational Learning Laboratory, CSLI, Stanford University, CA, Tech. Rep., 2006.

[16] D. Vernon, G. G. Metta, and G. Sandini, "A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 151–180, 2007.

[17] J. Anderson, D. Bothell, M. Byrne, S. Douglass, L. C, and Y. Qin, "An integrated theory of the mind," *Psychological Review*, vol. 111, pp. 1036–1060, 2004.

[18] J. Laird, A. Newell, and P. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial Intelligence*, vol. 33, pp. 1–64, 1987.

[19] T. Kohonen, *Self-Organisation and associative memory*. Springer, 1989.

[20] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[21] S. Hutchinson and A. Kak, "Extending the classical AI planning paradigm to robotic assemblyplanning," in *IEEE Int. Conf on Robotics and Automation*, 1990.

[22] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning About Objects Through Action - Initial Steps Towards Artificial Cognition," in *IEEE Int. Conf on Robotics and Automation*, 2003, pp. 3140–3145.

[23] S. Harnad, "The symbol grounding problem," *Physica*, vol. D, no. 42, pp. 335–346, 1990.

[24] L. Barsalou, "Grounded cognition," *Annual Review of Psychology*, vol. 59, pp. 617–645, 2008.

[25] M. Lungarella and O. Sporns, "Mapping information flow in sensorimotor networks," *PLoS Computational Biology*, pp. 1301–1312, 2006.

[26] A. Milner and M. Goodale, "Separate visual pathways for perception and action," *Trends in Neuroscience*, vol. 15, pp. 20–25, 1992.
[27] S. Glover, "Separate visual representations in the planning and control of action," *Behavioral and Brain Sciencest*, vol. 27, pp. 3–78, 2004.