Linköping University Post Print

Exploratory learning structures in artificial cognitive systems

Michael Felsberg, Johan Wiklund and Gösta Granlund

N.B.: When citing this work, cite the original article.

Original Publication:

Michael Felsberg, Johan Wiklund and Gösta Granlund, Exploratory learning structures in artificial cognitive systems, 2009, Image and Vision Computing, (27), 11, 1671-1687.

<http://dx.doi.org/10.1016/j.imavis.2009.02.012>

Copyright: Elsevier Science B.V., Amsterdam.

<http://www.elsevier.com/>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-21198>

Exploratory Learning Structures in Artificial Cognitive Systems^{*}

Michael Felsberg^{*}

Johan Wiklund

Gösta Granlund

Computer Vision Laboratory, Dept. EE, Linköping University, Sweden

Abstract

The major goal of the COSPAL project is to develop an artificial cognitive system architecture, with the ability to autonomously extend its capabilities. Exploratory learning is one strategy that allows an extension of competences as provided by the environment of the system. Whereas classical learning methods aim at best for a parametric generalization, i.e., concluding from a number of samples of a problem class to the problem class itself, exploration aims at applying acquired competences to a new problem class, and to apply generalization on a conceptual level, resulting in new models. Incremental or online learning is a crucial requirement to perform exploratory learning.

In the COSPAL project, we mainly investigate reinforcement-type learning methods for exploratory learning, and in this paper we focus on the organization of cognitive systems for efficient operation. Learning is used over the entire system. It is organized in the form of four nested loops, where the outermost loop reflects the user-reinforcement-feedback loop, the intermediate two loops switch between different solution modes at symbolic respectively sub-symbolic level, and the innermost loop performs the acquired competences in terms of perception-action cycles. We present a system diagram which explains this process in more detail.

We discuss the learning strategy in terms of learning scenarios provided by the user. This interaction between user ('teacher') and system is a major difference to classical robotics systems, where the system designer places his world model into the system. We believe that this is the key to extendable robust system behavior and successful interaction of humans and artificial cognitive systems.

We furthermore address the issue of bootstrapping the system, and, in particular, the visual recognition module. We give some more in-depth details about our recognition method and how feedback from higher levels is implemented. The described system is however work in progress and no final results are available yet. The available preliminary results that we have achieved so far, clearly point towards a successful proof of the architecture concept.

1 Introduction

Artificial Cognitive Systems (ACS) will become a future key technology with high impact on economy, society, and daily life. Autonomous robots with perceptual capabilities can be applied in many areas such as production, public and private service, science, etc.

The purpose of a cognitive system is to produce a response to appropriate percepts. The response may be a direct physical *action* which may change the state of the system or its environment. The response may be delayed in the form of a reconfiguration of internal models in relation to the interpreted *context* of the system. Or it may be to generate in a subsequent step a generalized *symbolic representation* which will allow its intentions to be communicated to some other system. As important as the percepts, is the dependence upon context.

A fundamental property of cognitive vision systems is that they shall be *extendable*. This requires that systems are able to both acquire and store information about the environment autonomously.

The views on vision architectures range between two extreme views [33]:

- Knowledge and scene representations must be supplied by the designer to the extent possible
- Knowledge about the external world has to be derived by the system through its own exploration

Proponents of the first view argue that if working systems are going to be available in some reasonable time, it is necessary to supply available information, and the modality for this is in declarative form. Proponents of the second view argue that if sufficiently complex models of the external world are going

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 215078 DIPLECS and from the European Community's Sixth Framework Programme (FP6/2003-2007) under grant agreement n° 004176 COSPAL.

* Corresponding author. Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, SE-58183 Linköping, Sweden

Email addresses: mfe@isy.liu.se (Michael Felsberg), jowi@isy.liu.se (Johan Wiklund), gosta@isy.liu.se (Gösta Granlund).

to be available to the system, it is necessary that it can explore and find these out essentially by itself.

We can assume that the economically optimal variety at the present time is somewhere between these extremes, such that the structural design ensures some efficiency in the performance. The present paper will deal with issues of the second view above.

Various alternatives for cognitive systems organization have been studied over the years. Important components are the development of hierarchical and distributed architectures [11,84,5,83,55,87,14,72,65,74]. Some of this started with the development of Active Vision [40,15,68,21] recognizing the fact that the system can and should modify its state of observation.

There is a general consensus that cognitive systems have to be embodied, as the loop involving actions is fundamental [69,81]. An interesting line of development is the use of simulations of the embodiment and external world, to train and bootstrap cognitive systems [7,32].

Other issues are to explore particular types of information representations, which may have advantages in allowing mode separations in large systems [12,34].

This paper is not intended to be any review of the state of the art of Cognitive Systems. Overviews can be found in [57,82]. Instead we focus on the background and techniques of the COSPAL architecture.

Much of the ideas of the COSPAL project are reflected in the paper [31] and many of the architectural ideas were motivated from the references cited earlier and from cognitive science, see e.g. [46], Chaps. 2 and 3, and cognitive neuroscience, see e.g. [29], Chap. 11. The philosophy behind the COSPAL project and the underlying ideas are elaborated in Section 2, which summarizes some of the ideas from [32].

Section 3 describes the technical implementation of the architecture which has been developed under the COSPAL project. Without going into technical methods developed during or applied in the project (see e.g. [25,47,26,23]) we concentrate on system aspects of the project in this paper.

Section 5 concludes the paper with some summarizing remarks and reflections.

In the subsequent discussion, there will be several references to known properties of biological vision systems. It should be emphasized at the outset that the ambition of this paper is not to argue possible models of biological vision systems, but to propose potentially effective architectures of technical systems. In this process, however, it is deemed useful to take hints from what is known about biological vision systems.

2 The Background of COSPAL

This section discusses organization aspects of cognitive systems, and proposes an architecture for cognitive systems. For general aspects of cognitive systems, we refer to the references given in the preceding section.

The inputs to a cognitive system, or the representations of information in early stages of it, are generally referred to as percepts. They will typically be visual or auditory, as these modalities generally carry most information about the environment. However, other sensing modalities may be used, in particular for boot-strapping or other support purposes. Perception and percepts are ambiguous terms, where some may say that perception is in fact the function performed by a cognitive system. However, there is generally agreement that percepts are compact, partially invariant entities representing the sensing space in question. Visual percepts will for example be some processed, more invariant, more compact representation of the information in an image, than the original iconic image obtained from the sensor. Examples are lines, edges, regions, etc.

A vision system receives a continuous barrage of percepts or input signals. It is clear that a system cannot attempt to relate every signal to all other signals. What mechanisms make it possible to select a suitable subset for processing, which will not drown the system? In the discussion of the COSPAL architecture, we focus on two major issues:

- **Action driven learning, or Action precedes Perception**
- **Relations between PA processing and Symbolic processing**

2.1 Perception-Action Learning

Over the years a number of articles have been written about Perception-Action (PA) issues [4,34,41]. Still, there is a great deal to add to this discourse.

What we see as the major mechanism for PA learning is:

Organization for learning is driven from the motor or action side and not from the sensing side.

Although this would appear like a simple principle, readers seem to miss a number of essential features of this, and components which make this workable for implementation. In addition, this principle is working at several levels of a system, in ways which may seem totally unrelated.

From biology we can get a clue for the organization at low levels, where semi-random signals are sent out from the motor system for organization of the nervous system. These generate muscle twitches at an early stage of development, which in turn activate sensors. This process, driven from the motor side, organizes the sensorial inputs of the nervous system. It has been convincingly shown that noise and spontaneously generated neural activity is an important component to enable organization of the visual system [51]. In a technical system, the base level that is hard-wired into the system, i.e., the perception-action capabilities that exist before learning, is rather arbitrary and can be related to what is known as primary reflexes in biological systems.

Although the particular motor signals may be semi-random, they are known to the system itself, and can then be associated to the sensor signals or percepts. This association may in a technical implementation imply the use of some textbook machine learning algorithm. It has been shown that control or knowledge of the motor signals is essential for the development of coordinated behavior. A passive sensing of the visual information is not sufficient to generate learning [39].

The PA part of the COSPAL system step by step performs a mapping from percepts onto actions or states. By states, we imply the physical position states of the system embodiment in some suitable geometric representation, certain internal processing states and certain external states of the objects and environment under attention. Actions imply changes of certain of these states, which are generated by the system.

The mapping from percepts onto actions or states implies an association of these or a learning process. It is important that this learning is performed incrementally, such that recently learned information can be used immediately.

2.1.1 A Continual Active-Reactive Learning Process

A system with this motor driven learning strategy will not simply just switch between two modes, where it is either learning, or in operation using its acquired knowledge. Rather it will simultaneously use both strategies, such that in the learning process, the system uses the competences it has acquired from earlier training experiences. In this way, increasingly complex PA primitives are built up, which is equivalent to the build-up of *concept spaces*, whose use will be discussed later [33].

We have earlier emphasized that learning takes place through exploration, and in that process action precedes perception. However, only a limited part of the process of the system is learning from the response-driven association, at a given instance. The remaining parts of the process will activate PA primitives, which have already been learned, and use these as building blocks. We will

use the following terminology:

By *self-generated action* we denote an activity of a system, which is produced without any apparent external influence. The action is assumed to be caused by a random noise signal affecting a choice or creating an activity in parts of the system.

By *reaction* we mean that the system performs an activity which is initiated or modified by a set of percepts. This is also referred to as percept-guided actions in the literature.

It should be noted that a particular action under consideration, will typically be composed by both self-generated components and reactive components. This is given an intuitive illustration in Figure 1.

In the training to deal with a certain phenomenon, here indicated at the highest level, the system associates percepts to the states which are changing. In this process, it uses earlier acquired information in a normal mapping of percepts onto actions in the lower levels. However, at the level to be trained, the action states constitute the organizing mechanism for the associations to implement.

Different parts in the structure use either of the two mechanisms at the same time:

- **Exploratory behavior:** Self-generated action $\rightarrow$ Percept $\rightarrow$ Association
- **Reactive behavior:** Percept $\rightarrow$ Reaction

A self-generated action by the system causes a particular set of percepts to appear. The actions and percepts are linked to each other in an association process.

This means that at any given time, most processes in a cognitive system implement a purposeful mapping of percepts onto actions according to the models acquired at training. However, there will generally be a (often highest) level of exploration, which is characterized by random action activities. These random components are propagated through the trained levels, where they perform purposefully within the domains of the experience acquired.

Similar ideas regarding the stepwise build-up of complex action primitives, have also been investigated by other authors, several references mentioned in the introduction, but also under the heading of Motor Babbling [17,18,13]. What is important in the COSPAL architecture is that the primitives semi-randomly activated are not just motor sequences, but increasingly complex PA sequences. This is very important for the convergence of an exploration/learning process, as it implies that the PA primitives generated, model phenomena at

Fig. 1. Intuitive illustration of hierarchical training procedure

some level in the external world, while the application in some particular context is subjected to a semirandom test.

Something we find important is that the exploratory/reactive mechanism, varieties of which have sometimes been denoted Babbling, seems to be universal, going from the lowest levels of neural organization to the highest levels of behavior. Complex PA primitives earlier built up, are applied semirandomly in different situations, where a successful application is memorized and incorporated in the repertoire of primitives. Since success is commonly indicated by some external feedback, this mechanism is nothing but what is known as reinforcement learning from the literature.

Important unresolved research questions are how these primitives are preselected to give a reasonable probability of success, in spite of their subsequent random application. The mixed exploratory/reactive strategy appears to be true even at high levels for cognitive systems of the category Humans: There is always a top level which deals with phenomena hitherto unknown to us and can only be subjected to exploration, implemented as pseudo-random activation of learned primitives at that level. These components are propagated down through the entire multi-level PA machinery acquired from experience, and implemented as valid action sequences for a sophisticated exploration of a new domain of the environment. So, the development of systems or biological organisms is a long sequence of babbling, although at increasingly higher levels of abstraction.

There are strong indications that the association of motor signals or actions and percepts, in the human brain, takes place in the posterior parietal cortex [45,67,71,88,59,75]. The principle of a combination between a probing into the external space and sensing, has recently received theoretical support [62].

2.1.2 Object-centered versus View-centered Representation

The PA cycle as discussed, turns out to be equivalent to a concept in computer vision: View centered object representation. The discussion in this paper of two different domains for processing, the PA domain and the symbolic domain, has an important relation to two different approaches to object representation: *view-centered* and *object-centered* representation, respectively [70,30]. See Figure 2.

Fig. 2. Object-centered and view-centered representation of an object. a) Measurements produce information about different views or aspects of an object. b) Object-centered representation: The views are used to reconstruct a closed form object representation. c) View-centered representation: The views are retained as entities which linked together form a representation of the object.

From a real object, a number of measurements or projections are generated. They may e.g. be images or other representations of the object, taken from different angles. See Figure 2a. From these measurements we can proceed along either one of two different tracks.

One of the tracks leads to the object-centered representation, which combines these measurement views into some closed form mathematical object [37]. See Figure 2b. The image appearance of an instance of a particular orientation of the object is then obtained using separate projection mappings.

A view-centered representation, on the other hand, combines a set of appearances of an object, without trying to make any closed form representation [80,63,73,8]. See Figure 2c.

Object-Centered Representation

The basic motive of the object-centered representation is to produce a representation which is as compact and as invariant as possible. It generally produces a closed form representation, which can subsequently be subjected to interpretation. This implies that no unnecessary information is included about details on how the information was derived. A central idea is that matching to a reference should be easier as the object description has no viewpoint-dependent properties. A particular view or appearance of the object can be generated using appropriate projection methods.

We can view the compact invariant representation of orientation as vectors and tensors [35], as a simple variety of object-centered representation. Over a window of a data set, a set of filters are applied producing a component vector of a certain dimensionality. The components of the vector tend to be correlated for phenomena of interest, which means that they span a lower dimensional sub-space. The components can consequently be mapped into some mathematical object of a lower dimensionality, to produce a more compact and invariant representation, i.e. a vector or a tensor [35].

A drawback of the object-centered representation is that it requires a preconceived notion about the object to ultimately find; its mathematical and representational structure, and how the observed percepts should be integrated to support the hypothesis of the postulated object. It requires that the expected types of relations are predefined and already existing in the system, and it requires an external system, which keeps track of the development of the system such as the allocation of storage, and the labeling of information. Such a preconceived structure is not well suited for self-organization and learning. It requires an external entity which can “observe labels and structure”, and take action on this observation. It is a more classical declarative representation, rather than a procedural representation.

View-Centered Representation

In a view-centered representation, no attempt is made to generalize the representation of the entire object into some closed form. The different parts are kept separate, but linked together using the states or responses, which correspond to or generated the particular views. This gives a representation which is not nearly as compact or invariant. However, it tells what state of the system is associated to a particular percept state. A view-centered representation in addition, has the advantage of being potentially self-organizing. This property will be shown to be crucial for the development of a learning PA structure. There are indications from perceptual experiments, that the view-centered representation is used in biological visual systems [70].

We can see that the view representation is structurally similar to the PA structure. The projection mappings mentioned correspond to the sets of percepts. The information about states under which these mappings are taken correspond directly to the actions and states of the PA structure. An important characteristic of the view representation is that it directly implements an *interpretation* (see Section 2.2), rather than a geometrical *description* of an object that we want to deal with. By interpretation we denote links to actions that are related to the object, and information about how the object transforms under the actions. This is what motivates the choice of structure for the PA unit described in earlier sections.

Combination of Representation Properties

An object centered representation is by design as invariant as possible with respect to contextual specificities. It has the stated advantage to be independent of observation angle, distance, etc. This has, however, the consequence that it cuts off all links that it has to specific contexts or response procedures which are related to a particular context or view. We recognize this from the next section as what is characterized as a *symbolic* representation. The generation of an invariant representation, implies discarding information which is essential for the system to act using the information. In order to use such information, a system has to introduce actual contextual information.

In parallel with the build-up of increasingly complex PA primitives, the structure will be of type frames-within-frames, where individual transformations of separate objects are allowed within a larger scene frame. This corresponds to the earlier derived PA primitives. It is postulated that the frames-within-frames partitioning is isomorphic with the state or response map structure. This implies that different action primitives are able to “attach” to the frame in question, to implement the PA invariance of a particular object aspect.

The crucially important reason for the efficiency of *action precedes perception* is that action or state space is much less complex than percept space. The number of possible combinations of perceptual primitives in an image is huge, and most combinations of these will not be of interest as they will never occur. It is necessary to fast identify the combinations which may occur, as economically as possible. Given that the state space is less complex, and that feature combinations of interest will be specifically apparent as the system moves around in the state space, this movement of the system in the state space can be used to organize the relevant parts of the feature space, and associate them to the generative states. This allows the system to separate an object from its background, separate distinct parts within an object, learn how the percepts transform under manipulation, etc. Actions can likewise be used to manipulate the environment, which in consequence will modify the emergent

percepts. Learning of these relations gives the system the information required for the subsequent use in the opposite direction: To use percepts to control actions in a flexible fashion.

The limited number of degrees of freedom allows the implementation of self-organizing or learning procedures, as there is a chance to perform fast converging optimizations. This gives a mechanism to step by step build up what finally may become a very complex processing structure, useful for the handling of a very complex niche of the external world. See also [33,30]. In contrast, it would never be possible to build up a complex processing structure, with a large number of degrees of freedom in a single learning or optimization phase.

To conclude, driving the system using response signals has four important functions:

- *To separate different modalities of an object from each other, such that they can be separately controlled.*
- *To identify the particular percepts which are related to a particular action modality*
- *To provide action outputs from the network generated. The action inputs during the associative learning, will later act bi-directionally as action outputs driven by matching percepts according to the learned relation.*
- *Related points in the response domain exhibit a much larger continuity, simplicity and closeness than related points in the percept domain. The response domain deals with physics, where e.g. occlusion (objects physically passing through each other) does not readily occur. In the percept domain occlusion often happens, and the occlusion point between two objects can in principle have a velocity exceeding the speed of light.*

Through active exploration of the environment, i.e. using PA learning, a system builds up *concept spaces*, defining the phenomena it can deal with in terms of PA primitives. Information can subsequently be acquired by the system within these concept spaces without interaction, by extrapolation using passive observation such as for imitation, or communication such as language. It is postulated that passively observed phenomena, for which no PA primitives have been acquired, can not be understood by the system.

2.2 Relations between PA processing and Symbolic processing

Much of the lack of success in machine vision for complex problems can be traced to the early view that percepts should generate a *description* of the object or the scene in question. There has traditionally been a belief that an abstraction of the scene should be generated as an intermediary step, before a response is synthesized. A great deal of the robotics field has been devoted

to the generation of such generalized descriptions [38], typically in geometric terms, with a conceptual similarity to CAD representations. See Figure 3. The ambition has often been that the model should describe the object geometry as accurately as possible. An extreme but common form of this is the statement that ... *the best model of an object is the object itself*. This represents a total misunderstanding of the core of the problem. An object does not carry with it any interpretation model. The fact that the current environment may be a convenient store for information is a different thing.

Fig. 3. Classical robotics model

In this description process of the image, objects shall be recognized and assigned to the proper categories, together with information about position and other relevant parameters. This description is then carried to a third unit, where the description is interpreted to generate appropriate actions into the physical world, e.g. to implement a robot. See Figure 3.

This structure has not worked out very well. In brief, it is because the leap between the abstract description and the action implementation is too large. A large amount of important quantitative information, of spatial and temporal nature necessary for precise action, has been lost in the abstraction process implicit in a description.

The major problem with this structure is that we primarily do not want a *description* of an object or a scene. What we need is an *interpretation*, which is not just a play with words but a very different entity, namely links between actions and states that are related to an object and corresponding changes of percepts. The purpose of cognitive vision systems is consequently not primarily to build up models of the geometry of objects or of scenes. It is rather to build up model structures which relate the percept domain and the action or state domain; to associate percepts emerging from an object, to states or actions performed upon the object.

To achieve this, it is necessary to eliminate the big leap between percept structure and action structure to form a more closely integrated structure, consisting of a sequence of smaller steps. In each such limited step, percepts and functions thereof are related at intermediary levels directly to corresponding states of the system itself or of the external world. This is exactly the PA structure discussed in the preceding section.

From all of this, one can conclude that the order between the later two parts should in fact rather be the *opposite*. See Figure 4.

Fig. 4. Perception-action robotics model

The unit for Scene description, which also includes symbolic processing, is now moved outside the tight and fast loop of feature analysis and action generation, but the units have a communication in both directions.

The next conceptual step in the presentation is to merge the first two units of Figure 4 into a single unit as indicated in Figure 5. The last unit in Figure 4 performing the abstract, descriptive and symbolic processing is now given a more appropriate labeling as indicated in Figure 5. A new unit is introduced as an interface between those mentioned earlier, to illustrate the importance of this function, which will be discussed later.

Fig. 5. Perception-action robotics model (extended)

This is the proposal for a more detailed structure for a technical cognitive vision system, as given in Section 3. We will first deal with a few more aspects.

2.2.1 Symbolic Representation and Processing

While the PA structure deals with the largely reactive *here-and-now* situation, the symbolic structure allows the system to deal with other points in space and time in an efficient way. This is what allows *generalization*. The symbolic processing structure deals with planning, language and communication. Symbolic representation and manipulation allows an efficient processing of abstract concepts in a relatively invariant format without unnecessary spatial, temporal and contextual qualifiers, which would severely complicate the processing. The invariant format makes manipulation and communication much more effective and its effects more generally applicable.

A symbolic representation is on the other hand a too meager form for sufficiently adaptive control of actions, a sometimes overlooked characteristic of

language. Human language works in spite of its relatively low information content, because it maps onto a rich spatial knowledge structure at all levels, available within our surprisingly similar brain structures. This information, however, derives from the individual exploration of what are similar environments.

The output from the symbolic processing is normally (for biological systems) output over the action output interface according to Figure 5. For a technical system, there may be possibilities to interface directly to the symbolic processing unit, as indicated by the dashed line to the right. The reason is that the symbolic representation used in this part of the system, can be expected to be less context dependent and consequently more invariant than what is the case in the PA part. This will make it potentially feasible to link symbol states to an outside representation, using association or learning.

2.2.2 Interface Between PA and Symbolic Representation

The interface between the PA and Symbolic processing units is an important part of the COSPAL system, which can in general terms be characterised as an interface between signal and symbol representation. There is a great deal of discussion around this issue [56,6,79,42,53], from which we will refrain to deal with in any detail.

For our purpose, the essential difference between a signal and a symbol is in terms of *invariance*. We do not see a clear boundary between what shall be denoted a signal or a symbol, as both will inevitably have several characteristics in common, such as their (potential) low level representation.¹ Rather we want to see representations as either more of a signal characteristic or more of a symbol characteristic, dependent upon their degree of invariance.

The processing in the PA part with its here-and-now orientation is very dependent upon quantitative contextual parameters, describing the system's position in space and other states in precise geometric terms. The transition from the PA representation to the symbolic representation implies in principle a *stripping off* of detailed spatial context. This transition produces sufficiently invariant packets of information to be handled as symbols or to be communicated.

After processing in the symbolic part, symbols will be fed back to the PA unit for, say, implementation of some action. This requires the reinsertion of current context with information about position etc., for the system to be able to implement the action properly. This mechanism allows *generalization*,

¹ Note that channel representations [34] can be used for representing signals as well as symbols.

in that the system can learn a procedure in one contextual setting, and apply this knowledge in a contextually different setting, as the current context is reinserted.

As a consequence, it is postulated that *metric* or quantitative similarity measures are only available within the PA part of a cognitive system and not in the symbolic part. This means that as two symbolic entities are to be compared, they are communicated from the symbolic part to the spatial-cognitive part of the system, where the comparison is implemented. One reason for this is that a relevant comparison usually requires the inclusion of current, quantitative parameters.

We can here observe a correspondence to the earlier discussion:

- **Representation in PA corresponds to View-Centered Representation**
- **Symbolic representation corresponds to Object-Centered Representation**

The output from a symbolic representation and manipulation is preferably viewed as designed for *communication*. This communication can be to another system, or to the PA part of the *own* system. This implies that the symbol structure is converted back to affect a fairly complex and detailed PA structure, where actual contextual and action parameters are reinserted in a way related to the current state of the system. In this way, symbolic information can be made to control the PA structure, by changing its context. The change of context may be overt or physical in commanding a different state or position, bringing in other percepts, or covert affecting the interpretation of percepts. The symbolic representation must consequently be translatable back to detailed contextual parameters relating to the current state of a system, and this translation must be made regardless if it is the own system or another system.

We believe that the subsequent symbolic representation shall emerge from, and be organized around, the action or state representation. This does not exclude the use of static clues such as color or any descriptive, geometric representation. There are strong indications that this is the way it is done in biological systems — it is known that our conscious perception of the external world is in terms of the actions we can perform with respect to it [76,28,77]. It has also been found that the perception of an object results in the generation of motor signals, irrespective of whether or not there is an intention to act upon the object [3,36].

This indicates how mechanisms such as learning from imitation can be implemented. If the "language" or representation used for the information into the symbolic unit is in terms of actions, it seems that learned perceived actions

of the own system, can more easily be generalized to the interpretation of the actions of another system. Actions imply some "normalization" to context, scale, objects, etc., which makes the action representation more invariant and better suited for symbolic processing. Strong links have also been discovered between brain mechanisms for language and action [66].

3 The COSPAL Architecture

In this section we describe the actually implemented system based on the ideas discussed before. We elaborate in some more detail the different parts of the architecture, the bootstrapping processes, and the achieved results.

3.1 An Implementation of the COSPAL Structure

The perception-action mapping part is attached to the the part for symbolic processing through an interface, which will be in the focus of this section. So far the discussion may have been interpreted that we would have a sharp division between a percept side and a response side in the structure. This is certainly not the case. There will be a mixture of percept and response components to various degrees in the structure. We will for that purpose define the notion of *percept equivalent* and *response equivalent*. A response equivalent signal may emerge from a fairly complex network structure, which itself comprises a combination of percept and response components to various degree. At low levels it may be an actual response muscle actuation signal which matches or complements the low level percept signal. At higher levels, the response complement will not be a simple muscle signal, but a very complex structure, which takes into account several response primitives in a particular sequence, as well as modifying percepts. The designation implies a complementary signal to match the percept signal at various levels. Such a complex response complement, which is in effect equivalent to the system state, is also what we refer to as *context*.

A response complement also has the property that an activation of it may *not necessarily* produce a response at the time, but rather an activation of particular substructures which will be necessary for the continued processing. It is also involved in knowledge acquisition and prediction, where it may not produce any output.

Note that the nested structure of feedback cycles in the architecture allow that certain classes of percepts may map onto actions after a very brief processing path, much like what we know as reflexes in biological systems. Such

direct mapped actions may however require a fairly complex contextual setup, which determines the particular mapping to use. Other actions may require a very complex processing, involving several levels of abstraction. This general, distributed form of organization is supported by findings in biological visual systems [88,78].

The number of levels involved in the generation of a response will depend on the type of stimulus input as well as of the particular input. In a comparison with biological systems, a short reflex arch from input to response may correspond to a skin touch sensor, which will act over interneurons in the spinal cord. A complex visual input may involve processing in several levels of the processing pyramid, equivalent to an involvement of the visual cortex in biological systems.

In Fig. 6 we give an overview of the COSPAL system architecture which is supposed to enable the above mentioned functionalities. The architecture consists of three major units, the Perception-Action (PA) unit for low-level control cycles, the Grounding & Management (GM) unit for managing and selecting models, and the Symbolic Processing (SP) unit for concept generation, supervision, and high-level control. Each unit has a simple control-feedback interface to its respective neighbors. The PA unit communicates also with the Hardware-layer and the SP unit through a User Interface with the user. The control and feedback signals are simple, in our current implementation even binary, and their purpose is to control the processing in the different modules. In this way, a pseudo-parallel processing of the different units becomes possible, see Algorithm 1.

Algorithm 1 Pseudo code for main loop.

```

1: loop
2:   HWfeedback = HW_poll(HWcontrol);
3:   [HWcontrol, PAfeedback] = PA_poll(PAcontrol, HWfeedback);
4:   [PAcontrol, GMfeedback] = GM_poll(GMcontrol, PAfeedback);
5:   [GMcontrol, SPfeedback] = SP_poll(SPcontrol, GMfeedback);
6:   SPcontrol = UI_poll(SPfeedback);
7: end loop

```

The information exchange takes place through shared state spaces. The simplest mechanism is between Hardware and PA unit: The Hardware unit writes the current percept into the state space and sends the feedback signal for a modified state space. The PA unit processes the percept, see lines 3–10 in Algorithm 2, writes an action into the state space (line 8), and sends a control signal to make the Hardware unit perform the action (line 9). This innermost loop runs at highest speed compared to the higher-level feedback-control loops.

The GM unit receives the state of the PA unit through the shared state space and maps it internally to existing models. As a result, a new state is generated

Fig. 6. System diagram of the COSPAL architecture.

Algorithm 2 Pseudo code for PA_poll (other poll functions read accordingly).

```
1: PAfeedback = 0;
2: HWcontrol = 0;
3: if HWfeedback then
4:   if PAmatch() then
5:     PAgenFeedback();
6:     PAfeedback = 1;
7:   else
8:     PAgenControl();
9:     HWcontrol = 1;
10:  end if
11: end if
12: if PAcontrol then
13:  PAupdate();
14:  PAgenControl();
15:  HWcontrol = 1;
16: end if
```

which is communicated to the PA unit. The PA unit itself tries to approach the in-fed state as close as possible by running the inner loop (lines 12–16). Whenever anything aside a simple approximation occurs (e.g. an internally unpredicted percept), it is communicated to the GM unit (lines 5–6).

The GM unit maps states onto symbolic representations by means of short-term memory slots and clustering methods. Whenever a state is not covered by the so-far acquired models, the GM unit initiates a new mode of the model and communicates this information to the SP unit. Uncovered states can result from, e.g., missing object models or from incomplete process models concerning the physical actuator. The SP unit can make use of this information to build new concepts in the object or process domain.

The SP unit also runs the post-generalization for categorizing object models, generalizing processes, and identifying goals. Based on these concepts, the SP unit can modify and restructure the symbolic part of the GM unit’s memory. Whereas the GM unit relates sub-symbolic to simple symbolic information, the SP unit relates symbols to each other and to symbolic tasks. The action sequences within symbolic tasks are also learned in the SP unit, based on perceived successful sequences. Again, this information is sent through the shared state space, and the control line just indicates the presence of new states.

Finally, the SP unit communicates to the user through the state space of the UI. The control and feedback signals here correspond to events in ordinary GUI programming, see Fig. 7 for the COSPAL GUI. Overall, the system uses its fourfold hierarchy of feedback-control loops in order to accomplish tasks of

different complexity in a robust way and in order to allow continuous learning over the whole lifetime.

Fig. 7. Screenshot of the COSPAL graphical user interface.

The separation of functionalities into three modules is motivated by the hierarchy of information and processes: sub-symbolic, first-order symbolic, and relations between symbols. In particular the learning that takes place in all three modules requires the distinction between these levels, as symbolic relations and mappings from sub-symbolic states to symbols must be trained hierarchically.

Since the initial system does not have any working models in its units, the output has to be generated according to some innate principle. The simplest thing to do is to generate random outputs and to perceive what the lower-level part returns. This mechanism has already been applied successfully at all levels, but some feedback or control from a higher level is always required. This feedback can be hard-wired, e.g., by defining a correct prediction of a state as success, typically applied at low levels of the system. The feedback might also be provided by propagating user feedback through the higher levels.

One key idea behind the nested structure of the COSPAL system is the feedback loops: each unit communicates with its respective neighboring units through feedback and control signals. We believe that it is a necessity to move the feedback and control to the respective neighboring levels in order to avoid pre-modeling of capabilities and representation. In a way, a single unit is not aware of its own performance, but the whole system is – up to the highest level where a user is required to give feedback. Of course the user feedback at highest level could also be replaced by learning from 'experience', but we consider this as providing the same information to the system in just another, but slower modality.

Without the interaction with the user, the system will never be shaped to

solve any problem. It is the user who specifies the task and provides the environment to the system: The user defines the learning scenario. It is also the user who assesses the system's performance at the highest level and reacts by providing further or different learning scenarios. In this way, the user helps the system building its models by providing learning problems at appropriate and increasing levels of difficulty.

3.2 Bootstrapping

Besides the continuous incremental learning mentioned above, the system is bootstrapped in the beginning in order to get it faster into a useful state. Although this distinction between two different modes may seem unnatural, this is also the case for biological systems. It is useful to replace extensive learning efforts on sparsely occurring events by intensive learning of massive occurrences in the beginning of model acquisition.

There is a fundamental difference between batch mode learning systems, i.e., a system which is either in learning mode or in operation mode, and systems with bootstrapping. In the former case, the system is exposed to two different states of the external world: in learning mode, typically stimuli and responses are given (supervised learning), whereas in operational mode only the stimuli are given, and hence, no further improvement of the system is possible. In the latter case, multiple stimuli and multiple feedback is present all the time, and it is the internal system state which distinguishes the two modes.

During the bootstrapping, the system follows a certain innate scheme for acquiring knowledge. We will illustrate this further below for the example of object recognition. This is motivated by certain behavioral schemes innate to, e.g., babies and small children: They try to reach what ever is in their range and they try to sense everything with their lips and tongue. One could postulate that this is a method to explore the dimensions of objects, textures, and shapes.

After bootstrapping, the system is controlled by its own supervisor part, established in the symbolic processing. This has the effect that the system appears to behave purpose driven. Only if processing fails at any level, the particular level undergoes a further learning step similar to those during the bootstrapping: It acquires a new solution to the problem by some strategy. The main difference to bootstrapping is that the system returns into a purpose driven behavior immediately after the new competence has been acquired.

In a more sophisticated scenario, one could think about setting only single levels or modules of levels into bootstrap mode. As a consequence, each module could stop bootstrapping independently. The overall system behavior seems

to gradually change from innate schematic to purpose driven likewise in a biological system. This is however not implemented in the current system design.

The technical implementation of bootstrapping and incremental learning differs between the three levels. For the symbolic level, bootstrapping is very much a symbol grounding problem, cf. [10,52]. The mechanisms applied therein are graph-based clustering techniques for percept-action couples, whereas actions are to be understood on a symbolic level. The percept representation used in the mentioned method is a pictorial one, as more generic ones were not available at that time.

To ground percept symbols on image features in a generic way is one task of the medium system layer. The technical implementation of its bootstrapping is based on advanced classification techniques in terms of binary classifiers, which are based on equivalences of pairs of objects. This technique is based on [27].

At the lowest level of the system, bootstrapping might be performed as supervised or unsupervised learning, c.f. [44,64] for servoing and clustering. Going beyond (unsupervised) feature clustering is difficult as the semantic correspondence between different learning instances is not obvious. In particular for bootstrapping visual recognition this leads to a number of dilemmas: How shall we learn to recognize something if we do not even know whether we see the same thing or a different one? How shall we generalize what we have learned if we do not know which parts of the input space are equivalent according to a yet unknown symbol? How can we avoid unwanted clustering of inputs if they are indistinguishable in the feature domain (example: extract a white ball lying on a white line)?

3.3 Example 1: Visual Bootstrapping and Learning of PA Cycles

Bootstrapping mechanisms have been applied at several different levels of the COSPAL system. For the symbolic level, an extensive overview has been published in [85]. In this section, we will focus on an examples for bootstrapping at the lower levels of the system: Bootstrapping visual recognition [49,50].

This bootstrapping problem can be illustrated by the experimental setup in Fig. 8, where a typical training view and a similar test view are shown. For visual recognition it is not sufficient to learn visual descriptors of objects, as the background might interfere with the object appearance. The common way to solve this problem is to segment the object first and then to recognize it. We believe that this should be done simultaneously instead, i.e., the proper segmentation is learned as a part of the object appearance.

Fig. 8. Left: training view (without background), right: test view (with cluttered background)

For this purpose, the training views need to be arranged such that they are effectively without background. This can be done as in Fig. 8 by creating artificially a background in a distinct color or by exploiting the variation with respect to the background. In the latter case 'without background' for the training view is implemented by achieving a very flat distribution of visual features over the whole background area. As a result of appropriate probabilistic treatment, the background in the test view does not influence the matching result.

But how do we achieve the necessary foreground-background segmentation if we do not know anything about the object? Exactly this issue is addressed by the perception-action learning paradigm. If we assume that the system pushes around objects with its manipulator, it becomes trivial to segment the object (moving part) against the background (static part).

One problem in this context is the manipulator itself: it might occlude or disturb the feature extraction and thus the prototype generation process. Currently we solve this issue by moving the manipulator out of the field of view. A more elegant and in particular much faster way is to make the recognition blind for the effects of the manipulator. This can be done in a similar way as with the background before. We first train the recognition of the manipulator which gives uniform distributions for the background, see Sect. 3.4. Then we virtually subtract the density of the manipulator from the current density estimate in order to become blind for the manipulator. This method has however not been fully evaluated yet.

In the way described so far, we can recognize previously seen objects at a known location. Two problems remain, where only the first one is related to bootstrapping: How do we detect candidate regions for the recognition? And: How can we generalize to previously unseen combinations of object properties? Both questions are relevant in the context of incremental learning and feedback mechanisms. Although this belongs no longer to the bootstrapping process

itself, we give a brief description for the sake of completeness.

Detection of recognition candidates has been suggested earlier in terms of interest point clusters, cf. [22]. The actual scheme for generating the interest maps can certainly be improved, but the starting point will always be: take the current view and generate a map of candidates. The candidates have different weights and those with the largest weights are chosen first. If those candidates do not lead to significant recognitions, the interest map is modified according to the distance to already tested points. This scheme can be used during bootstrapping and during incremental learning. In the latter case, the upper levels can even actively influence the map in a coarse way: they provide a prior to the detection result.

Another interesting problem is the process of generalization during incremental learning. Given the following situation: we have bootstrapped a red round object, a green round object, a red cube, and a yellow cube. Is it possible to use the bootstrapped recognition system to recognize, e.g., a green cube - without knowing that color lives in a certain feature subspace and shape in another one? Preliminary results seem to confirm this, but the learning time grows to evolutionary orders of magnitude.

Thus, it appears to be impractical not to consider shape and color as independent measurement dimensions. The low-level detectors for both type of features are commonly separated, as orientation information is typically extracted from derivatives of the intensity, whereas colors are extracted as area properties (e.g. blobs). In the further processing a separation of shape and color information allows the generalization to objects with different combinations of already learned color and shape features by exploiting the separability of the two dimensions.

Note that even though color and shape are considered to be uncorrelated on the PA level, it is still possible to suppress or enforce certain combinations of color and shape at later levels of processing. The difference to considering the joined space of color and shape from the beginning is that the PA part will first recognize an (incorrect) object hypothesis (e.g. a blue banana), which will at symbolic level be rejected (banana are never blue). Still, this is much more efficient than requiring the correct feature combination already at the PA level.

Once that the object recognition works on the separable subspace level, i.e., different properties can be synthesized in arbitrary combinations, recognition results can be directly attached to symbolic representations. The mapping between subspace property prototypes and sets of symbols can be bootstrapped directly, without modifying single symbols and properties individually. This becomes possible by correspondence-free association of symbols to subspaces

by means of a learning technique proposed in [48]. This results in an efficient scheme for bootstrapping direct links to the symbolic level.

3.4 Example 2: Learning of PA Cycles

In this second example for bootstrapping, we give a brief overview of our instantiation of learning robot control [54]. In the cited work, we describe a way to learn PA-based robot control using three conceptional contributions:

- visual recognition of the manipulator by a generic patch-voting method
- learning the inverse kinematic by linear models (LWPR)
- refinement of positioning by exploiting visual recognition and the linear models in a feedback-loop (closed loop visual servoing)

For the visual recognition, a similar methodology as the one in Sect. 3.3 has been applied, but with two differences:

- In contrast to the objects that we want to grasp, the robot is not rigid but it is assumed to consist of a number of rigid elements.
- We have direct and active control over the robot arm.

The first difference makes the recognition more difficult, the second one makes the learning much easier. The robot can be moved entirely out of the field of view, allowing an efficient background subtraction during learning. The gripper can easily be detecting by opening and closing it.

For the inverse kinematics and the camera control of the system, basically two options exist:

- Measure all geometric information of the setup (e.g. length of limbs, position of joints, position of cameras, camera parameters) as accurate as possible and derive analytic expressions for all required mappings.
- Learn all required mappings by moving the robot and observing what is happening in the camera views.

The first alternative is potentially much more accurate than the second one, but it suffers from an inflexibility against minimal distortions in the setup, e.g. if the cameras are slightly displaced. For an industrial environment this might be acceptable, but not for artificial cognitive systems.

The drawback of learning mappings between the visual domain and the control domain is the lack of accuracy due to the relatively crude approximations that have to be applied on this high-dimensional learning problem. Hence, open-loop visual control of the manipulator becomes impossible. Instead, visual

servoing, i.e., closed-loop control is required to compensate for the approximation errors in the learned kinematics.

The advantage of a learning-based approach is the higher degree of robustness concerning changes of the setting. If, for instance, the cameras are moved, the increased error in the visual servoing can be used to trigger some further learning steps on the fly, resulting in an adaptive inverse kinematics. Convergence of the servoing will be slower in the beginning but after a few iterations, the system is back at the old performance [43].

The absolute accuracy that can be achieved with such a strategy stays of course below industrial standards, but is still sufficient to, for instance, insert objects into holes. In combination with force-feedback manipulators, the accuracy is sufficient to fulfill a large variety of tasks that are typical for the human level of manipulation accuracy.

4 The COSPAL System Implementation

The system has been implemented based on the ideas and techniques that are described in previous sections. As an overview paper, we do not aim at covering all technical aspects of the system in detail, but instead we gave some examples, references, and summaries above. In this section, we now give an overview of the system performance at the end of the project, the technical and engineering tricks that had to be applied at some levels of the system to get it to work, and finally we will give an overview of all learning methods that have been used in context of COSPAL systems, i.e., the main demonstrator and other system studies.

4.1 *Experimental Evaluation*

Once all required methods had been integrated into a common system using the algorithmic frame corresponding to the structure in Fig. 6, we have been able to evaluate that the predictions made about the architecture are correct.

4.1.1 *The Complexity Chain*

In order to measure the progress with respect to the overall project goals, it has earlier been suggested to formulate a complexity chain [61], which the integrated system has to run through. In Tabel 1, we present the final status of the chain for the final COSPAL demonstrator together with the respective motivation for each part. This chain is the result of intensive discussions within

the COSPAL consortium and it describes the kinds of problems that the final demonstrator system has to solve.

Table 1

Complexity chain of the COSPAL demonstrator

step	motivation
1. bootstrap visual recognition for all shapes & colors by pushing around	show the absence of modeling is replaced with exploratory learning
2. one bootstrapped object in the scene	show success of recognition
3. imitate example 'approach object and grasp' instructed by teacher through GUI	transfer action to own system
4. imitate with unknown position	generalize movements
5. random exploration for picking object and placing into hole (50% chance of trying to pick the hole)	deficiency in current representation property: grippability learning hole detection
6. add distractor: flat block	extend association of visual propert.
7. add new shape	extend symbolic representation
8. move the board	show spatial invariance

4.1.2 Limitations of Scenario

For practical reasons, we were forced to put some limitations onto the scenario. We give the motivation for the respective limitations below. They are:

- the puzzle holes are located in the ground plane
- reference object-to-hole positioning is obtained from physical arrangement of object in the hole
- objects must be oriented such that they can be rotated to hole orientation in one move

Ground plane restriction This restriction allows us to avoid a second side-view camera. As all objects can only lie in a single plane, the mapping from image coordinates to 3D coordinates is uniquely defined. This mapping (a homography) is however not modeled, but learned. In the same way, one could learn a mapping from two cameras to unique positions (up to pathologic cases) in a full 3D setting. However, we believe that this would just add technical complexity to the system instead of proving conceptually different properties. Therefore, we selected the single side-view scenario.

Object-to-hole reference Since our artificial system would destroy the puzzle if we would let it explore the object-to-hole relation freely, i.e., let it trying to insert randomly, we had to replace the random exploration with reinforcement feedback with positive examples, i.e., supervised learning. Given that occurrence of positive examples is guaranteed, the results from the learning are the same. In the 2D simulator, we have shown that this is actually the case [22].

Orientation restriction This restriction is implied by the physical restrictions of the manipulator. The manipulator cannot rotate to arbitrary state, which implies that an arbitrary object-to-hole constellation can only be solved if the object is re-gripped in between. Again, omitting this restriction would add a lot of technical complexity without proving conceptual new properties. On the symbolic level, it has been shown that similar actions as re-gripping can be learned.

4.1.3 Achieved Results

The system successfully goes through the entire complexity chain. It learns to solve the puzzle and it behaves robustly. We have tested robustness by changing the background, putting objects out of reach, changing the board, changing the configuration during processing, and by moving the cameras. We have run the system through nearly three entire days for the subsequent results.

The system is fairly accurate, such that about 80% of the insertion actions are successful. Grasping objects never failed due to inaccuracies, only due to failure in visual detection. Nothing about the concrete task is hard-coded into the system, such that exactly the same system can be applied to, for instance, stacking coins according to their value. This has however not been tested.

The speed of processing has not been a relevant issue in the COSPAL project. Nevertheless we tried to achieve a processing speed that allows several evaluations of the whole complexity chain per hour, i.e., the system moves slightly slower than an about one-year old infant that is trying to solve the same puzzle. The servoing steps are done every five seconds and the entire cycle of recognition, approaching, aligning, and grasping an object takes about 90 seconds.

4.2 Overview: Applied Learning Methods in the System Demonstrators

In the final system implementations, i.e., the main demonstrator and the other demonstrators, many different methods for machine learning have been ap-

plied. Those methods known from the literature, e.g. [9], are mentioned without further reference.

For the visual servoing in the main system, hierarchical networks of locally arranged models (HLAM, [43]), a variant of a network of experts, has been developed and used. For learning to imitate human arm movements (extra demonstrator), dynamic cell structures [1] and reinforcement (Q) learning have been used [2].

For the visual servoing of a low-cost robotic arm (extra demonstrator), local linear models trained by LWPR [54] have been used. The visual recognition within the main demonstrator makes use of a novel variant of prototype learning which at the same time locally optimizes control parameters [49]. For the matching criterion different metrics for comparing densities (Kullback-Leibler divergence, Cramér-von-Mises distance) have been used [24].

For the visual recognition of textured puzzles (extra demonstrator), a variant of AdaBoost, called Waldboost have been used [58]. The latter combines AdaBoost with Wald’s sequential probability ratio tests. Data has been organized in a decision tree which was organized similar to CART [16]. For the clustering of recognized hypotheses and mapping to pose parameters, support vector machines and regression techniques have been used [20].

On the symbolic level, unsupervised logic-inference-based perceptual re-parameterization has been applied as well as supervised stochastic perceptual re-parameterization [86]. Unsupervised and online clustering methods have been applied for learning distances by observation [60]. Boosting and hybrid methods have been applied for automatically modeling and partitioning the appearance space of objects on the fly with no a-priori learning [19].

4.3 *Reflections on Engineering Artificial Cognitive Systems*

According to previous Sects. 2–3, the COSPAL system implementation tries to replace modeling with learning to a largest possible extent. The motivation behind this is to improve robustness and adaptability of the system to previously unforeseen situations. We believe that this is the only way to build artificial cognitive systems, as the real world is too complex to specify exactly and completely the task space of a cognitive system.

As a consequence of this design decision, the system development cannot be based on methods from software modeling, which require to specify exact prerequisite or post-conditions. The popular *Unified Modeling Language* (UML) is based on *Use Cases*, which require pre-condition and post-condition specifications. A learning system should, in general, improve its ability to solve a certain task if it has been exposed to learning samples for the task. The task-

space is only restricted by the input and the output modalities of the system. Hence, the typical Use Case for a learning system will remain too unspecific to be really useful: Learning samples are provided to the system. As a consequence, the system changes its states in order to improve its performance according to some pre-defined criterion.

As such a Use Case is rather meaningless for the system design, we decided to follow another strategy, namely to break down the design according to the learning tasks of different complexity, see Sect. 4.1.1. By structuring the requirements for the learning module according to the tasks in the complexity chain, one can abstract from the concrete problem to the different types of learning problems. Once the learning system is implemented, one can verify whether it reacts to the concrete learning problem as expected. As a proof of concept, we have successfully applied a sub-system to an entirely different learning task that the original shape sorter puzzle: a radio controlled car.

What remains to be decided on the base of rather classical design principles, are those system parts that are related to physical sensors and actuators. Therefore, we had to decide whether to accept certain limitations, see Sect. 4.1.2, or to model a way to circumvent them. One example for the latter one is that we divided the low-level perception-action cycles into four different steps, due to the limited movability of cameras and the pure physical strength of the used robotic arm. Our system does not use a single vision system, but two different cameras: side-view and end-effector camera, see Fig. 9. Instead

Fig. 9. Left: A typical shape-sorter puzzle. Right: the COSPAL demonstrator setup with manipulator (RX 90, left), an end-effector camera attached to it, and the side-view camera.

of continuously varying the scale of processing, the system switches between two different discrete scales. Furthermore, grasp and release actions had to be separated from the other movements in order not to destroy the setup in the case of wrong positioning. Altogether, four different system states have been defined:

approach: The system uses the side-view camera to move the manipulator to roughly the right place.

align: The system uses the end-effector camera to move the manipulator into a position to grasp the object.

grasp: The manipulator is lowered in two steps, asking the user whether to continue in-between. The gripper is closed.

release: The manipulator is lowered in two steps, asking the user whether to continue in-between. The gripper is opened.

4.4 The COSPAL Api

The following api (application program interface) implements the four different system states described above:

success = approach() The position of the targeted object in the side-view camera image is extracted from the object hypotheses. This position is used to move the end effector to the corresponding (learned) position in the scene.

success = align() Implements a visual servoing loop where the current end effector view is compared with trained views, and corresponding difference in robot pose is updated.

success = grab() Implements the gripper down, close gripper, gripper up by calling the corresponding sequence of functions in the robot api (see below).

success = release() Implements the gripper down, open gripper, gripper up by calling the corresponding sequence of functions in the robot api (see below).

The following api to communicate with the robot and cameras was implemented:

getStaticImage() Returns an image from the side-view camera.

getEndeffectorImage() Returns an image from the end-effector camera.

rotateAroundZAxis(roll) Rotates the end effector around the axis perpendicular to the ground plane.

driveInXYPlane([x y]) Translates the end effector parallel to the ground plane.

[x y z yaw pitch roll] = getRobotPosition() Returns the robot position state.

getGripperStatus() Returns the gripper state: 0=closed, 1=open, 2=holding object.

openGripper() Opens the gripper.

closeGripper() Closes the gripper.

driveToLowerPlane() Lower the end effector to a position just above the ground plane.

driveToUpperPlane() Raises the end effector to a higher position from the

ground plane.

Fig. 10. A view from the (static) side-view camera.

Figure 10 shows a view from the static camera. The correspondence between the positions of the objects in the image, and the robot position with the gripper roughly on top of the corresponding object is learnt. This mapping is dynamic, i.e., if the static camera is moved slightly, the system adapts and updates the mapping dynamically.

Fig. 11. Left: A view from the end-effector camera. The empty gripper is positioned at roughly the right place. Right: A view after aligning to a grippable position using image-based visual servoing.

When the end effector is placed roughly in the right position, i.e. after the approach, the end effector is aligned to a position from which the object can be gripped or released. Figures 11, 12 show examples of views before and after the alignment procedure.

Fig. 12. Left: A view from the end-effector camera. The gripper, holding an object, is positioned roughly on top of a hole. Right: A view after aligning to a position from where the object can be released into the hole. The alignment of the object in the gripper to the hole is based on visual servoing.

4.5 *Functionality of the Units*

In Table 2, we list the functionalities implemented in the system units from Fig. 6 for realizing the demonstrator. A snapshot of the running system is depicted in Fig. 13, illustrating the functionality of all units.

Table 2

Functionalities implemented in the respective units (FSM = final state machine).

System unit	module
SP	fixed set of actions + FSMs by imitation
SP	learn distances
SP	automatic generation FSM
SP	hierarchical active exploration
GM	temporal book-keeping of object hypotheses
GM	inter-camera object recognition fusion
PA	incremental learning for visual servoing (side-view camera)
PA	gripper alignment (end-effector camera view)
PA	gripping
PA	visual recognition using channel-coded feature maps
PA	bootstrapping of object detection and recognition module (side-view camera)
PA	bootstrapping of view interpolation module (end-effector camera)

Since we focused mostly on the PA unit so far, we give some illustrations of the GM unit in Fig. 14. The GM unit clusters votes from the PA module for different kinds of objects and filters out intermediate hypotheses.

Fig. 13. Screenshot showing views of all processing levels in the working system. Upper left: Object hypotheses generation (PA). Upper right: Object classification (GM). Lower left: Action planning, “move blue halfcircular object and put it in the corresponding hole”, indicated by blue arrow (SP).

Fig. 14. Left: View from side-view camera with classified object hypotheses overlaid. Right: View from end-effector camera with classified object hypotheses overlaid.

The supervision process of the SP unit is illustrated by the layered feedback model in Fig. 15. Each attempt to a certain action that fails is repeated a number of times until negative feedback is provided to the next higher level. At that level, the same process is repeated, i.e., the higher-level command is issued once more.

Fig. 15. Feedback cycles in the low-level perception-action loops. The structure is the same for grasping and releasing.

Consider for example the case that grasping is not performed since the user replied not to continue, aka grasping failed. As a consequence, aligning is repeated. If grasping continues to fail, aligning eventually responds failure, thus making the system to repeat the approach command. After the new approach, alignment is repeated and if repeatedly failing, approach is re-issued. If approach is also failing repeatedly, the symbolic part will get notice of that and chooses another object. This is the essential mechanism that is applied repeatedly in random exploration, i.e., in the case where the symbolic part has no evidence of which object is the right one to choose.

5 Concluding Remarks

There is a traditional belief that percepts are in some way “understood” in a vision system, after which suitable responses are attached. This does however require simple units to have an ability of “understanding”, which is not a reasonable demand upon local structures. This is a consequence of the luxury of our own capability of consciousness and verbal logical thinking. This leads to a potential danger inherent in our own process for system development, in that our conscious experience of the environment has its locus at the very end of the chain of neural processing; at a point where in fact most of the cognitive processing has already been made. It is well established that language capability, logical reasoning and conscious processing are all derivatives of the motor system normally in the left half of the brain, at the end of the processing chain.

A processing in terms of such conscious object terms *inside* the cognitive processing structure is not likely to occur. Most, or nearly all of the cognitive processing has already been done when signals are available for the motor manipulation objects to take place, or a representation in conscious terms to emerge. We are too late to see *how* it happened. We only notice *what* has happened.

Given the well known distributed nature of processing, it is apparent that there is no basis for any estimation of importance or “meaning” of percepts locally in a network, but that “blind and functional rules” have to be at work to produce what is a synergic, effective mechanism. One of these basic rules is undoubtedly to register how percepts are associated with responses, and the consequences of these. This seems at first like a very limited repertoire, which could not possibly give the rich behavior necessary for intelligent systems. Like in the case of other evolutionary processes for self-organization, there seems to be no other credible alternative. Rather, we have to look for simple and robust rules, which can be compounded into sufficient complexity to deal with complex problems in a “blind” but effective way.

In most of the preceding discussion, a feedback perception-action structure has been assumed, which primarily reminds of robotics applications, as it implies an embodiment which can interact with the external world. Does that mean that the preceding cognitive vision architecture is only applicable to robotics? *Not exclusively!* The belief is however that the structure discussed is not only advantageous, but inevitable, for the development of demanding applications of vision. This also includes the interpretation of complex static imagery.

It will similarly be inevitable for cases where the output is not a physical action but the communication of a message. An example of the latter type is complex man-machine interfaces, where the actions and speech of a human are registered, interpreted and communicated symbolically to a system to implement a very sophisticated control of its functions. Sufficient flexibility and adaptation requires learning for the system to deal with all contextual variations encountered in practical situations.

The training of cognitive systems for such advanced but non-robotic applications requires the development of mixed real-virtual training environments. In these, the system will gradually build up its knowledge of its environment with objects including humans. The learning is again implemented as association, between the learning systems own state parameters, and the impinging perceptual parameters. The typical case is as discussed earlier that the system moves an object in front of its camera input. The movement parameters are known to the system and can be associated with the percepts appearing as results of the movements. This can in training environments be simulated in various ways such that corresponding state and percept information is made available to the system.

In such a way, competence can be built up step by step, in a mixture of real and virtual training environments. With a design allowing incremental learning, it shall be possible to start out with reasonably crude virtual environments, to give the system some tentative knowledge of object and environment space structure, which is refined in the real environment.

From this derives the view that the development of powerful technical vision systems inevitably has to go the path over perception-action mapping and learning, similarly to the case for robotics, even if the systems will be used for interpretation of static imagery or to generate and communicate messages to other systems. This opens up wide ranges of applications of cognitive systems at an early stage, which do not require advanced mechanical manipulators. One such important application field is in activity interpretation for man-machine interfaces. Setting up training environments for such applications is one of the research challenges in cognitive vision.

Acknowledgment

We would like to point out that all of this work is based on the close co-operation with our project partners in our partner labs in Kiel, Prague, and Surrey. We furthermore acknowledge the support and valuable comments of our project officer Hans-Georg Stork and our project reviewers Murat Kunt and Lucas Paletta.

References

- [1] I. Ahrns, J. Bruske, G. Sommer, On-line learning with dynamic cell structures, in: Proc. of ICANN 95, Vol. 2, 1995.
- [2] S. Al-Zubi, G. Sommer, Imitation learning and transferring of human movement and hand grabbing to adapt to environment changes, in: B. R. Reinhard Klette, Dimitris Metaxas (ed.), Human Motion - Understanding, Modeling, Capture and Animation. 13th Workshop, Lecture notes in Computer Science, Springer, Schloss Dagstuhl, Germany, 2006.
- [3] S. J. Anderson, N. Yamagishi, V. Karavia, Attentional Processes link perception and action, Proc R Soc Lond B Biol Sci. 269 (1497) (2002) 1225–1232.
- [4] J. Antonisse, H. Wechsler, Integrating perception, action and learning, SIGART Bull. 2 (4) (1991) 25–28.
- [5] R. C. Arkin, An Behavior-based Robotics, MIT Press, Cambridge, MA, USA, 1998.
- [6] R. Bajcsy, The problem of signal and symbol integration: A study of cooperative mobile autonomous agent behaviors, in: In Proceedings of KI-95, LNCS, Springer, 1995.
- [7] E. Bar Aviv, E. Rivlin, Functional 3d object classification using simulation of embodied agent, in: BMVC, 2006.

- [8] D. Beymer, T. Poggio, Image Representations for Visual Learning, *Science* 272 (1996) 1905–1909.
- [9] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.
- [10] R. Bowden, L. Ellis, J. Kittler, M. Shevchenko, D. Windridge, Unsupervised symbol grounding and cognitive bootstrapping in cognitive vision, in: *Proc. 13th Int. Conference on Image Analysis and Processing*, 2005.
- [11] R. A. Brooks, A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* 2 (10).
- [12] R. A. Brooks, Intelligence without representation, *Artificial Intelligence* 47 (1991) 139–159.
- [13] D. Caligiore, T. Ferrauto, D. Parisi, N. Accornero, M. Capozza, G. Baldassarre, Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping, in: *International Conference on Cognitive Systems (CogSys2008)*, Karlsruhe, Germany, 2008.
- [14] A. Chella, M. Frixione, S. Gaglio, A cognitive architecture for artificial vision, *Artificial Intelligence* 89 (1–2) (1997) 73–111.
- [15] H. Christensen, K. Bowyer, H. Bunke, Active robot vision, in: *Active Robot Vision*, 1993.
- [16] O. Chum, J. Matas, Geometric hashing with local affine frames, in: A. Fitzgibbon, C. Taylor, Y. LeCun (eds.), *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, IEEE Computer Society, Los Alamitos, USA, 2006.
- [17] Y. Demiris, A. Dearden, From motor babbling to hierarchical learning by imitation: a robot developmental pathway (2005).
URL <http://cogprints.org/4961/>
- [18] R. Der, G. Martius, From motor babbling to purposive actions: Emerging self-exploration in a dynamical systems approach to early robot development, in: *From Animals to Animats 9, 9th International Conference on Simulation of Adaptive Behavior, SAB 2006*, Rome, Italy, September 25–29, 2006, *Proceedings, Lecture Notes in Computer Science*, Springer, 2006.
- [19] L. Ellis, R. Bowden, Learning responses to visual stimuli: A generic approach, in: *5th International Conference on Computer Vision Systems*, Bielefeld, Germany, 2007.
- [20] L. Ellis, R. Bowden, Linear predictors for fast simultaneous modeling and tracking, in: *submitted to Workshop on Non-rigid Registration and Tracking through Learning, Eleventh IEEE Intl. Conf. Computer Vision*, Rio de Janeiro, Brazil, 2007.
- [21] J. A. Fayman, E. Rivlin, H. I. Christensen, Av-shell, an environment for autonomous robotic applications using active vision, *Auton. Robots* 6 (1) (1999) 21–38.

- [22] M. Felsberg, P.-E. Forssén, A. Moe, G. Granlund, A COSPAL subsystem: Solving a shape-sorter puzzle, in: AAAI Fall Symposium: From Reactive to Anticipatory Cognitive Embedded Systems, Crystal City, Arlington, Virginia USA, 2005.
- [23] M. Felsberg, P.-E. Forssén, H. Scharr, Channel smoothing: Efficient robust smoothing of low-level signal features, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2) (2006) 209–222.
- [24] M. Felsberg, J. Hedborg, Real-time view-based pose recognition and interpolation for tracking initialization, *Journal of Real-Time Image Processing* 2 (2–3) (2007) 103–116.
- [25] P.-E. Forssén, B. Johansson, G. Granlund, Channel associative networks for multiple valued mappings, in: 2nd International Cognitive Vision Workshop, Graz, Austria, 2006.
- [26] P.-E. Forssén, A. Moe, Autonomous learning of object appearances using colour contour frames, in: 3rd Canadian Conference on Computer and Robot Vision, IEEE Computer Society, Québec City, Québec, Canada, 2006.
- [27] V. Franc, V. Hlaváč, A novel algorithm for learning support vector machines with structured output spaces, Research Report K333–22/06, CTU–CMP–2006–04, Department of Cybernetics, Faculty of Electrical Engineering Czech Technical University, Prague, Czech Republic (May 2006).
- [28] M. S. Gazzaniga, *The Social Brain. Discovering the Networks of the Mind*, Basic Books, New York, 1985.
- [29] M. S. Gazzaniga, R. B. Ivry, G. R. Mangun, *Cognitive Neuroscience*, 2nd ed., W. W. Norton & Company, New York, 2002.
- [30] G. Granlund, Does Vision Inevitably Have to be Active?, in: Proceedings of the 11th Scandinavian Conference on Image Analysis, SCIA, Kangerlussuaq, Greenland, 1999, also as Technical Report LiTH-ISY-R-2247.
- [31] G. Granlund, A Cognitive Vision Architecture Integrating Neural Networks with Symbolic Processing, *Künstliche Intelligenz* (2) (2005) 18–24, ISSN 0933-1875, Böttcher IT Verlag, Bremen, Germany.
- [32] G. Granlund, Organization of Architectures for Cognitive Vision Systems, in: H.-H. Nagel (ed.), *Cognitive Vision Systems*, Springer, 2005.
- [33] G. H. Granlund, The complexity of vision, *Signal Processing* 74 (1) (1999) 101–126, invited paper.
- [34] G. H. Granlund, An Associative Perception-Action Structure Using a Localized Space Variant Information Representation, in: Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC), Kiel, Germany, 2000.
- [35] G. H. Granlund, H. Knutsson, *Signal Processing for Computer Vision*, Kluwer Academic Publishers, 1995, ISBN 0-7923-9530-1.

- [36] J. Grezes, M. Tucker, J. Armony, R. Ellis, R. E. Passingham, Objects automatically potentiate action: an fmri study of implicit processing, *Eur J Neurosci* 17 (12) (2003) 2735–2735.
URL <http://www.blackwell-synergy.com/links/doi/10.1046/j.1460-9568.2003.02695.x/abs>
- [37] W. E. L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, MA. USA, 1990.
- [38] R. M. Haralick, L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, 1992.
- [39] R. Held, A. Hein, Movement-produced stimulation in the development of visually guided behavior, *Journal of Comparative and Physiological Psychology* 56 (5) (1963) 872–876.
- [40] J. Herve, Y. Aloimonos, Exploratory active vision: Theory, in: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Urbana Champaign, (USA), 1992.
- [41] B. Hommel, J. Müsseler, G. Aschersleben, W. Prinz, The Theory of Event Coding (TEC): A framework for perception and action planning, *Behavioral and Brain Sciences* 24 (05) (2002) 849–878.
- [42] P. Hong, T. Huang, Learning to extract temporal signal patterns from temporal signal sequence, in: *Proceedings of the 15th International Conference on Pattern Recognition (ICPR'00)*, 2000.
- [43] F. Hoppe, Local learning for visual robotic systems, Ph.D. thesis, Christian-Albrechts-Universität zu Kiel, Institut für Informatik (to submitted, 2007).
- [44] F. Hoppe, G. Sommer, Fusion algorithm for locally arranged linear models, in: *Proc. of 18th Int. Conf. on Pattern Recognition (ICPR)*, 2006.
URL <http://doi.ieeecomputersociety.org/10.1109/ICPR.2006.590>
- [45] M. Jeannerod, *The Cognitive Neuroscience of Action*, Blackwell publishers Ltd, 1997.
- [46] M. Jeannerod, *Motor Cognition*, Oxford Psychology Series, Oxford University Press, 2006.
- [47] B. Johansson, T. Elfving, V. Kozlov, Y. Censor, P.-E. Forssén, G. Granlund, The application of an oblique-projected landweber method to a model of supervised learning, *Mathematical and Computer Modelling* 43 (2006) 892–909.
- [48] E. Jonsson, M. Felsberg, Correspondence-free associative learning, in: *18th International Conference on Pattern Recognition (ICPR'06)*, 2006.
- [49] E. Jonsson, M. Felsberg, Accurate interpolation in appearance-based pose estimation, in: *Proc. 15th Scandinavian Conference on Image Analysis*, vol. 4522 of LNCS, 2007.

- [50] E. Jonsson, M. Felsberg, Efficient computation of channel-coded feature maps through piecewise polynomials, *Image and Vision Computing*.
- [51] L. C. Katz, C. J. Shatz, Synaptic Activity and the Construction of Cortical Circuits, *Science* 274 (1996) 1133–1138.
- [52] J. Kittler, M. Shevchenko, D. Windridge, Visual bootstrapping for unsupervised symbol grounding, in: J. Blanc-Talon, W. Philips, D. Popescu, P. Scheunders (eds.), *Proceedings of 8th Advanced Concepts for Intelligent Vision Systems International Conference*, Springer, 2006.
- [53] N. Krüger, Three dilemmas of signal- and symbol-based representations in computer vision, in: *BVAI*, 2005.
- [54] F. Larsson, E. Jonsson, M. Felsberg, Visual servoing for floppy robots using LWPR, in: *RoboMat*, 2007, accepted.
- [55] T. S. Lee, D. Mumford, Hierarchical Bayesian inference in the visual cortex, *Optical Society of America* 20 (7) (2003) 1434–1448.
- [56] M. Levine, Special section on signals and symbols, *Proceedings of the IEEE* 84 (11) (1996) 1623–1704.
- [57] M. Lungarella, G. Metta, R. Pfeifer, G. Sandini, Developmental robotics: a survey, *Connection Science* (2003) 151–190.
- [58] J. Matas, J. Šochman, Wald’s sequential analysis for time-constrained vision problems, in: S. Hutchinson (ed.), *IEEE International Conference on Robotics and Automation, Workshops and Tutorials*, IEEE, 2007, cDROM, invited paper.
- [59] K. Matsumoto, W. Suzuki, K. Tanaka, Neuronal Correlates of Goal-Based Motor Selection in the Prefrontal Cortex, *Science* 301 (5630) (2003) 229–232.
URL <http://www.sciencemag.org/cgi/content/abstract/301/5630/229>
- [60] E. Ong, R. Bowden, Learning distances for arbitrary visual features, in: *Proc. of British Machine Vision Conference 2006*, vol. 2, BMVA Press, 2006.
- [61] E. Ong, L. Ellis, R. Bowden, Learning through imitation, *Image and Vision Computing* Submitted to the same issue.
- [62] D. Philipona, J. O’Regan, J.-P. Nadal, Is There Something Out There? Inferring Space from Sensorimotor Dependencies, *Neural Comp.* 15 (9) (2003) 2029–2049.
URL <http://neco.mitpress.org/cgi/content/abstract/15/9/2029>
- [63] T. Poggio, S. Edelman, A network that learns to recognize three-dimensional objects, *Nature* 343 (1990) 263–266.
- [64] H. Prehn, G. Sommer, An adaptive classification algorithm using robust incremental clustering, in: *18th International Conference on Pattern Recognition, ICPR 2006*, August 20–24, Hong-Kong, vol. 1, 2006.
- [65] T. J. Prescott, P. Redgrave, K. Gurney, Layered control architectures in robots and vertebrates, *Adaptive Behavior* 7 (1999) 99–127.

- [66] F. Pulvermüller, Brain mechanisms linking language and action, *Nature Reviews Neuroscience* 6 (7) (2005) 576–582.
URL <http://dx.doi.org/10.1038/nrn1706%0D%0A>
- [67] J. Quintana, J. M. Fuster, From Perception to Action: Temporal Integrative Functions of Prefrontal and Parietal Neurons, *Cereb. Cortex* 9 (3) (1999) 213–221.
URL <http://cercor.oupjournals.org/cgi/content/abstract/9/3/213>
- [68] R. P. N. Rao, D. H. Ballard, An active vision architecture based on iconic representations, Tech. Rep. TR548, University of Rochester (1995).
- [69] A. Riegler, When is a cognitive system embodied?, *Cognitive Systems Research* 3 (3) (2002) 339–348.
URL [http://dx.doi.org/10.1016/S1389-0417\(02\)00046-3](http://dx.doi.org/10.1016/S1389-0417(02)00046-3)
- [70] M. Riesenhuber, T. Poggio, Computational models of object recognition in cortex: A review, Tech. Rep. 1695, Artificial Intelligence Laboratory and Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology (Aug. 2000).
- [71] E. Robertson, J. Tormos, F. Maeda, A. Pascual-Leone, The Role of the Dorsolateral Prefrontal Cortex during Sequence Learning is Specific for Spatial Information, *Cereb. Cortex* 11 (7) (2001) 628–635.
URL <http://cercor.oupjournals.org/cgi/content/abstract/11/7/628>
- [72] R. Salomon, Improving the dac architecture by using proprioceptive sensors, in: In SAB98, 1998.
- [73] B. Scholkopf, H. A. Mallot, View-Based Cognitive Mapping and Path Planning, *Adaptive Behavior* 3 (3) (1995) 311–348.
URL <http://adb.sagepub.com/cgi/content/abstract/3/3/311>
- [74] D. Schrer, H. Burchardt, F. Sill, F. Golasowski, R. Salomon, D. Timmermann, Role-based strategies at example of robocup., in: M. H. Hamza (ed.), *Artificial Intelligence and Applications*, IASTED/ACTA Press, 2005.
URL <http://dblp.uni-trier.de/db/conf/aia/aia2005.html#SchroerBSGST05>
- [75] A. B. Schwartz, D. W. Moran, G. A. Reina, Differential Representation of Perception and Action in the Frontal Cortex, *Science* 303 (5656) (2004) 380–383.
URL <http://www.sciencemag.org/cgi/content/abstract/303/5656/380>
- [76] R. W. Sperry, *Science and Moral Priority: Merging Mind, Brain and Human Values*, Praeger, New York, 1985.
- [77] S. P. Springer, G. Deutsch, *Left Brain, Right Brain*, Freeman, New York, 1993.
- [78] G. Sumbre, Y. Gutfreund, G. Fiorito, T. Flash, B. Hochner, Control of Octopus Arm Extension by a Peripheral Motor Program, *Science* 293 (5536) (2001) 1845–1848.
URL <http://www.sciencemag.org/cgi/content/abstract/293/5536/1845>

- [79] A. Teller, M. Veloso, Evolutionary learning for orchestration of a signal-to-symbol mapper, in: Image Understanding Workshop, 1997.
- [80] S. Ullman, R. Basri, Recognition by linear combinations of models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13 (10) (1991) 992–1006.
- [81] D. Vernon, Cognitive vision: The case for embodied perception, *Image and Vision Computing* 26 (1) (2008) 127–140.
- [82] D. Vernon, 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 (2) (2007) 151–179.
- [83] P. F. M. J. Verschure, Distributed adaptive control: Explorations in robotics and the biology of learning, *Informatik/Informatique* 1 (1998) 25–29.
- [84] P. F. M. J. Verschure, B. J. A. Kröse, R. Pfeifer, Distributed adaptive control: The self-organization of structured behavior, *Robotics and Autonomous Systems* 9 (3) (1992) 181–196.
URL <http://dblp.uni-trier.de/db/journals/ras/ras9.html#VerschureKP92>
- [85] D. Windridge, J. Kittler, Cognitive bootstrapping: A survey of bootstrap mechanisms for emergent cognition in natural and artificial agents, *ACM Computer Surveys* (submitted).
- [86] D. Windridge, J. Kittler, Open-Ended Inference of Relational Representations in the COSPAL Perception-Action Architecture, in: *Proc. of International Conference on Machine Vision Applications (ICVS 2007)*, Bielefeld, Germany, 2007.
- [87] M. Wooldridge, N. R. Jennings, Intelligent agents: Theory and practice, *The Knowledge Engineering Review* 10 (2) (1995) 115–152.
- [88] U. Ziemann, Sensory-motor integration in human motor cortex at the pre-motoneurone level: beyond the age of simple MEP measurements, *J Physiol (Lond)* 534 (3) (2001) 625–.
URL <http://jp.physoc.org/cgi/content/abstract/534/3/625>