Exploratory Learning Structure in Artificial Cognitive Systems*

Michael Felsberg, Johan Wiklund, Erik Jonsson, Anders Moe, Gösta Granlund

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-58183 Linköping, Sweden
email mfe@isy.liu.se

Abstract. One major goal of the COSPAL project is to develop an artificial cognitive system architecture with the capability of exploratory learning. Exploratory learning is a strategy that allows to apply generalization on a conceptual level, resulting in an extension of competences. Whereas classical learning methods aim at best possible 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. Incremental or online learning is an inherent requirement to perform exploratory learning.
Exploratory learning requires new theoretic tools and new algorithms. In the COSPAL project, we mainly investigate reinforcement-type learning methods for exploratory learning and in this paper we focus on its algorithmic aspect. Learning is performed in terms 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 most existing 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

The COSPAL project[1] aims at developing a new architecture for artificial cognitive systems. This architecture combines techniques from the areas of artificial neural networks and artificial intelligence. At all levels of the system, perception-action learning is applied for training. The learning is incremental and exploratory, i.e., the system can continuously extends its capabilities [1]. Much of the ideas of the COSPAL project are reflected in the paper [2] and many of

* This work has been supported by EC Grant IST-2003-004176 COSPAL. This paper does not represent the opinion of the European Community, and the European Community is not responsible for any use which may be made of its contents.

[1] http://www.cospal.org

Proceedings of the ICVS International Cognitive Vision Workshop – ICVW 2007
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

the architectural ideas were motivated from cognitive science, see e.g. [3], Chaps. 2 and 3, and cognitive neuroscience, see e.g. [4], Chap. 11.

Without going into technical methods developed during or applied in the project (see e.g. [5–8]) we concentrate on system aspects of the project in this paper. First results in this direction have been published in [9], where a simulated shape-sorter puzzle is solved by an early instance of a COSPAL system, c.f. Fig. 1.

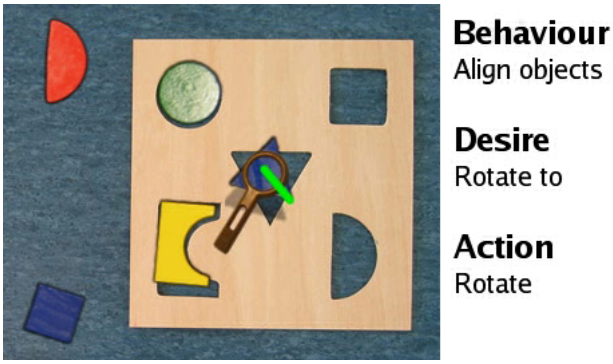

Fig. 1. An early version of the COSPAL system solving a simulated shape-sorter puzzle.

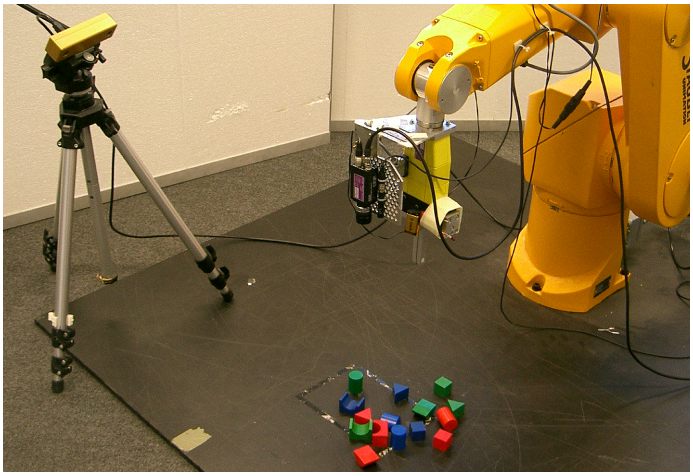

Fig. 2. Real 3D shape-sorter puzzle.

In the final demonstrator of the project, the COSPAL system will learn to solve a real 3D shape-sorter puzzle, c.f. Fig. 2. The important difference to existing systems accomplishing similar tasks is that we do not model objects and the scene in terms of hard-coded structures which are then specifically connected to the visual percepts. Instead, our system is supposed to build models from the appearance of objects and the exploration of the environment with

Proceedings of the ICVS International Cognitive Vision Workshop – ICVW 2007
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

the manipulator. This starts already with learning the appearance of the own manipulator, thus leading to the capability of controlling movements in the environment. As a second step the system builds concepts of objects by observing the possibility to manipulate parts of the environment. Learning to solve tasks with these acquired object models is a further step, where largest part of the activity is moved to higher-level processing.

An essential part of the higher-level processing is the communication and interaction with the user. This interaction supports the system in its activities of generalization, categorization, process modeling, and goal identification. The user serves as a final controlling instance, which shapes the system top-down. The main difference to proposed exploratory systems in the literature, see e.g. [10], is the layered architecture which allows feedback loops within the system.

2 The COSPAL Architecture

In Fig. 3 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 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.

Proceedings of the ICVS International Cognitive Vision Workshop – ICVW 2007
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

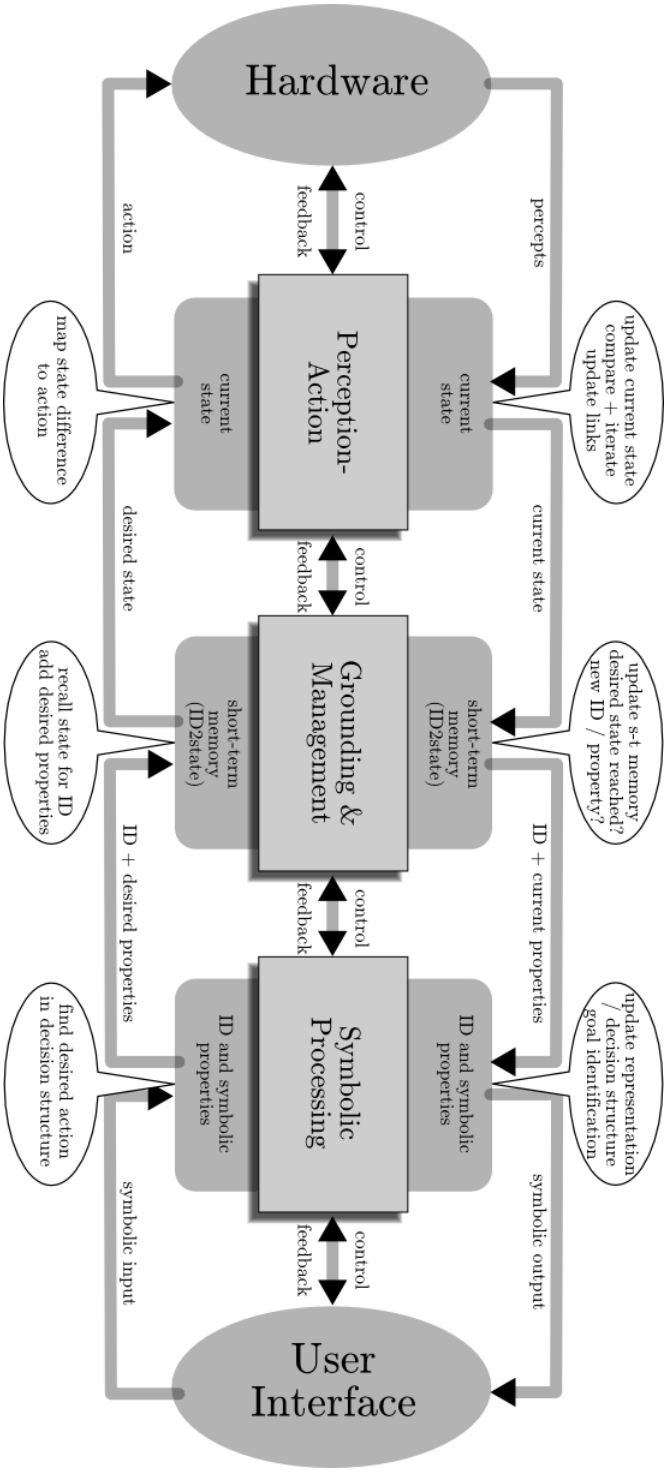

Fig. 3. System diagram of the COSPAL architecture.

Proceedings of the ICVS International Cognitive Vision Workshop – ICVW 2007
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

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

```
PAfeedback = 0;
HWcontrol = 0;
if HWfeedback then
   if PAmatch() then
      PAgenFeedback();
      PAfeedback = 1;
   else
      PAgenControl();
      HWcontrol = 1;
   end if
end if
if PAcontrol then
   PAupdate();
   PAgenControl();
   HWcontrol = 1;
end if
```

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. 4 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.

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 places 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 are the feedback loops: each unit communicates with its respective neighbored units through feedback and control signals. We believe that it is a necessity to move the feedback and control to the respective neighbored levels in order to avoid pre-modelling of capabilities and representation. In a way, a single unit is not

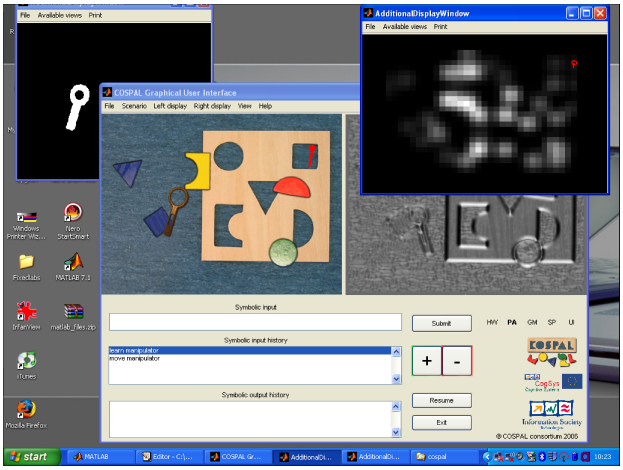

Fig. 4. Screenshot of the COSPAL graphical user interface.

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 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 levels of difficulty.

3 Bootstrapping

Besides the continuous incremental learning mentioned above, the system is bootstrapped in the beginning in order to get it quicker 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 systems 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 realized 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. [11, 12]. 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 [13] and allows to identify subspaces of the input-space which remain equivalent for different training samples with certain common properties, e.g., color or shape.

At the lowest level of the system, bootstrapping might be performed as supervised or unsupervised learning, c.f. [14, 15] 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)?

4 Bootstrapping and Learning in the PA Part: New Techniques

The visual recognition module is based on the representation technique described in [16]. As it has been shown in [8] that this class or representations are equivalent to a sampled kernel density estimator. The obvious way to compare such elements in this representation is in terms of the cross entropy or Kullback-Leibler divergence:

$$D(\mathbf{p}, \mathbf{q}) = \sum_j p_j \log \frac{p_j}{q_j} \ , \tag{1}$$

where $\mathbf{p}$ and $\mathbf{q}$ are the representations of a prototype and a query, respectively. The visual recognition module selects a number of image features, computes their sampled density estimates, and compares those with a number of prototype densities.

This recognition technique results in receiver operator characteristics which are more or less perfect for simple scenarios as the COIL-100 database (12 training views, 60 test views). The integral under the ROC curve is one (rounded to two digits) [17]. The interesting property of the method is that even in case of missing information and cluttered background nearly the same recognition rates can be achieved. In Fig. 5, a typical training view and a similar test view are shown.

Proceedings of the ICVS International Cognitive Vision Workshop – ICVW 2007
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

Fig. 5. Left: training view (without background), right: test view (with cluttered background)

'Without background' for the training view is realized as a uniform density over the whole background area. As a result, the background in the test view does not influence the value of the cross entropy (1). 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 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. 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.

Detection of recognition candidates has been suggested earlier in terms of interest point clusters, cf. [9]. 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.

Even more interesting is the problem 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, if we make use of the earlier mentioned binary classifier at the medium layer.

The idea is as follows: During bootstrapping, the binary classifier is trained according to the output of the visual recognition, i.e., recognition of the first four objects. If the binary classifier is trained, it will notice that object 1 and 2 have something in common, object 3 and 4, and object 1 and 3. In this way, the binary classifier learns to split the features into two subspaces which humans would call color and shape. The problem is now that binary classifiers can only be trained on very limited feature spaces, whereas the described visual recognition uses some thousand of dimensions.

Assuming that the reduced feature set for the classificator is obtained by a linear projection of the full feature set: $\mathbf{f} = \mathbf{PF}$, a reasonable thing to do is to find the subspace division in the full feature space from the subspace division in the reduced feature set. A minimum norm approach makes sense here, as it tries to avoid large variations and lead to equally filled subspaces. If $\mathbf{sf}$ is the subspace of the reduced feature space, we minimize the norm of the subspace projections for all full features $\mathbf{F}_k$ under the constraint that the projection of the subspace equals the subspace of the projection:

$$\min_{\mathbf{S}} \sum_k \|\mathbf{SF}_k\|^2 \qquad \text{s.t.} \qquad \mathbf{sP} = \mathbf{PS} \ . \tag{2}$$

The subspace projection $\mathbf{SF}$ can then be used to verify separate properties of objects, e.g. color or shape. The advantage of this method is that the separation of features has not been modeled into the system, such that it works for arbitrary selections of features without changing the code. To allow successful recognition, the features need to be sufficiently rich of course.

Once that the object recognition works even on subspace level, i.e., different properties can be synthesized in arbitrary combinations, they 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 [18]. This results in an efficient scheme for bootstrapping direct links to the symbolic level.

5 Ongoing Work and Outlook

It should be pointed out, that we are still integrating some of the mentioned methods into the algorithmic frame corresponding to the structure in Fig. 3. This means that the reported results are obtained by single modules or off-line exchange of data and not yet in a closed system. Although we are constantly improving modules, not all code has yet reached the same level as it is described here and future publications including exhaustive experiments will follow.

Future work will concentrate on the partially open topics named in the previous section, as there are:

- Virtually removing the manipulator
- Improving the candidate detection
- Evaluating the subspace projection method

New results will be available soon and are going to be presented on a workshop in June, held with the SCIA conference.

Acknowledgment

We would like to point out that all of this work is based on the close cooperation with our project partners in our partner labs in Kiel, Prague, and Surrey.

References

1. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning **38**(3) (2000) 257–286
2. Granlund, G.: A Cognitive Vision Architecture Integrating Neural Networks with Symbolic Processing. Künstliche Intelligenz (2) (2005) 18–24 ISSN 0933-1875, Böttcher IT Verlag, Bremen, Germany.
3. Jeannerod, M.: Motor Cognition. Oxford Psychology Series. Oxford University Press (2006)
4. Gazzaniga, M.S., Ivry, R.B., Mangun, G.R.: Cognitive Neuroscience. 2nd edn. W. W. Norton & Company, New York (2002)
5. Forssén, P.E., Johansson, B., Granlund, G.: Channel associative networks for multiple valued mappings. In: 2nd International Cognitive Vision Workshop, Graz, Austria (May 2006) 4–11
6. Johansson, B., Elfving, T., Kozlov, V., Censor, Y., Forssén, P.E., Granlund, G.: The application of an oblique-projected landweber method to a model of supervised learning. Mathematical and Computer Modelling **43** (2006) 892–909
7. Forssén, P.E., Moe, A.: Autonomous learning of object appearances using colour contour frames. In: 3rd Canadian Conference on Computer and Robot Vision, Québec City, Québec, Canada, IEEE Computer Society (June 2006)
8. Felsberg, M., Forssén, P.E., Scharr, H.: Channel smoothing: Efficient robust smoothing of low-level signal features. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(2) (2006) 209–222
9. Felsberg, M., Forssén, P.E., Moe, A., Granlund, G.: 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)
10. Christiansen, A.D., Mason, M., Mitchell, T.: Learning reliable manipulation strategies without initial physical models. In: IEEE International Conference on Robotics and Automation. Volume 2. (May 1990) 1224 – 1230
11. Bowden, R., Ellis, L., Kittler, J., Shevchenko, M., Windridge, D.: Unsupervised symbol grounding and cognitive bootstrapping in cognitive vision. In: Proc. 13th Int. Conference on Image Analysis and Processing. (September 2005) 27–36
12. Kittler, J., Shevchenko, M., Windridge, D.: Visual bootstrapping for unsupervised symbol grounding. In: Proceedings of 8th Advanced Concepts for Intelligent Vision Systems International Conference, Springer (September 2006) 1037–1046
13. Franc, V., Hlaváč, V.: 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)
14. Hoppe, F., Sommer, G.: Fusion algorithm for locally arranged linear models. In: 18th International Conference on Pattern Recognition (ICPR'06). Volume III. (2006) 1208–1211
15. Prehn, H., Sommer, G.: An adaptive classification algorithm using robust incremental clustering. In: 18th International Conference on Pattern Recognition (ICPR'06). Volume I. (2006) 896–899 to appear.
16. Felsberg, M., Granlund, G.: P-channels: Robust multivariate m-estimation of large datasets. In: 18th International Conference on Pattern Recognition (ICPR'06). (2006)
17. Felsberg, M., Hedborg, J.: Real-time visual recognition of objects and scenes using p-channel matching. In: Proceedings SCIA. (2007)
18. Jonsson, E., Felsberg, M.: Correspondence-free associative learning. In: 18th International Conference on Pattern Recognition (ICPR'06). (2006)