A COSPAL Subsystem: Solving a Shape-Sorter Puzzle*

Michael Felsberg and **Per-Erik Forssén** and **Anders Moe** and **Gösta Granlund**
Computer Vision Laboratory,
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

To program a robot to solve a simple shape-sorter puzzle is trivial. To devise a Cognitive System Architecture, which allows the system to find out by itself how to go about a solution, is less than trivial.

The development of such an architecture is one of the aims of the COSPAL project, leading to new techniques in vision based Artificial Cognitive Systems, which allow the development of robust systems for real dynamic environments. The systems developed under the project itself remain however in simplified scenarios, likewise the shape-sorter problem described in the present paper.

The key property of the described system is its robustness. Since we apply association strategies of local features, the system behaves robustly under a wide range of distortions, as occlusion, colour and intensity changes. The segmentation step which is applied in many systems known from literature is replaced with local associations and view-based hypothesis validation. The hypotheses used in our system are based on the anticipated state of the visual percepts. This state replaces explicit modeling of shapes. The current state is chosen by a voting system and verified against the true visual percepts. The anticipated state is obtained from the association to the manipulator actions, where reinforcement learning replaces the explicit calculation of actions. These three differences to classical schemes allow the design of a much more generic and flexible system with a high level of robustness.

On the technical side, the channel representation of information and associative learning in terms of the channel learning architecture are essential ingredients for the system. It is the properties of locality, smoothness, and non-negativity which make these techniques suitable for this kind of application. The paper gives brief descriptions of how different system parts have been implemented and show some examples from our tests.

*This work has been supported by EC Grant IST-2003-004176 COSPAL. It reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein.

Introduction

In the COSPAL project (The COSPAL consortium 2004), we try to overcome the limitations of today's Artificial Cognitive Systems (ACS) by developing a new system design and architecture. The long-term perspective beyond COSPAL is to design fully autonomous systems for dynamic real-world scenarios. To achieve this goal, we believe it is necessary to first solve the problem of controlled real-world scenarios in which the ACS is the only agent, which is the objective of COSPAL. To our knowledge, there exist no really robust solutions to this simplified problem yet.

Robustness is a very important requirement of a working ACS. The common definition of robustness for ACS is to perform reasonable even for unforeseen circumstances (Vernon 2004). Hence, every system design step which contains models of the world, its objects, their relations, or their behaviour potentially reduces robustness, since all these models can only represent a subset of possible instantiations.

Therefore, our strategy in the COSPAL project is to avoid modelling as much as possible and to replace it with action-driven associations which are learned by the system (Granlund 2003). This cannot be achieved in a single, huge effort, but must be built-up in a step-wise way. As a consequence, we have to first reconsider even simpler problems than controlled real world scenarios and focus on simulated environments and actors first. This type of problem can easily be solved by model-based systems, but the interesting aspect in this paper is to show that this is also possible with a system based on associations.

The system which is presented in this paper consists of a learning-based algorithm which interacts with a simulated puzzle, see figure 1. The simulator uses bitmaps of real objects, but the images are rendered using a generator based on a parameter set for the objects. The simulated manipulator allows actions like rotate and move, grip and release object.

The aim is now to make a simple ACS learn to recognise (identify and localise) the objects and holes, to align them, and thereby to solve the puzzle. In the present system we use a simple rule-based module for the high-level part of the system, but one of the later project goals in COSPAL is to replace them with advanced symbolic reasoning methods for goal-directed planning. The focus of this paper is, however, on the low-level and mid-level parts of the system.

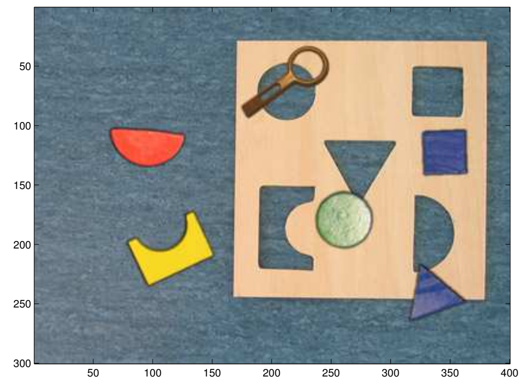


Figure 1: Image from puzzle simulator. The object at the top, centre is the manipulator.

A Simple COSPAL System

The system currently has four different behaviours, which are *innate*, or hard-wired. For later instances of the system we plan to replace these schemes to a large extent with symbolic learning. For the current implementation, the four behavioral schemes are:

B1 learn manipulator-action-appearance map
B2 find object
B3 learn object appearance
B4 align objects.

The system learns about its environment in behaviours **B1**-**B3**, and applies the acquired knowledge in **B4**. As it can be seen from the state graph in figure 2, behaviours **B1**-**B3** have a common structure which establishes a action-perception/prediction-learning (APL) loop.

The top three boxes constitute **B1**. In this behaviour, the system learns to predict how the appearance of the manipulator changes as it is moved. This allows the system to find out what parts of the scene constitute the manipulator, and then avoid its attention to be drawn to it.

The three boxes in the middle of figure 2 below form **B2**. This is essentially an attention switching mechanism, which moves the manipulator between potentially interesting (not recognised) locations.

The three boxes at the bottom form **B3**, in which the system learns how an object changes appearance by manipulating it. After the system has run out of new objects to explore it switches to **B4**, and solves the puzzle.

Figure 3 illustrates the first part of the recognition system from a computational view. The 2D puzzle generator renders puzzle images and if in learning mode, it also provides the current parameters of the manipulator. This is how the system learns to predict the manipulator appearance. Learning of object appearances is done in a similar way, with the notable exception that no object parameters are provided by the simulator. We will in the following sections describe the system in more detail.

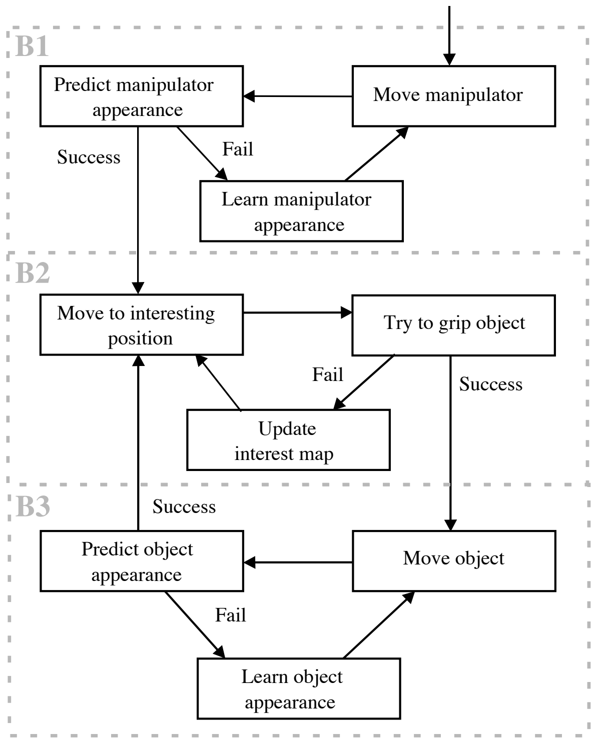


Figure 2: State graph for COSPAL system behaviour.

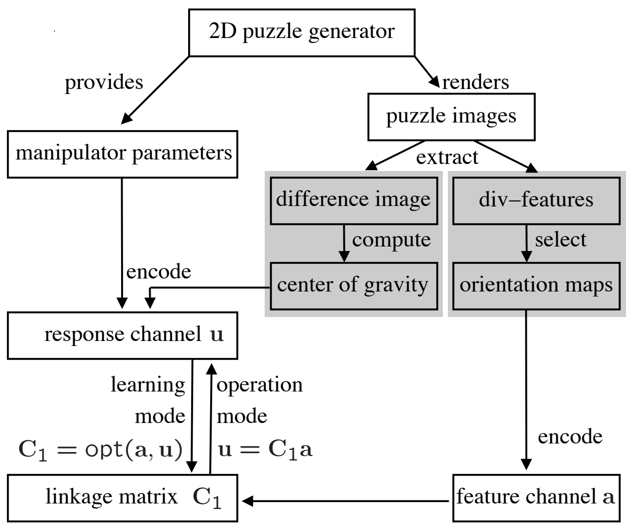


Figure 3: Recognition part 1 of our COSPAL subsystem. The grey boxes preprocess the visual percepts and are different for different levels of the system. Note further that for **B2** no optimization is performed in our implementation. Instead, the interest map is multiplied with a decay field.

Interest Points and Orientation Maps

From the rendered images, the system extracts local *interest points* (IP) or foci of attention for the recognition step. In the present implementation we use local maxima of a first order rotational symmetry map, i.e., div-features (Johansson & Granlund 2000), and the map's direction at the maxima.

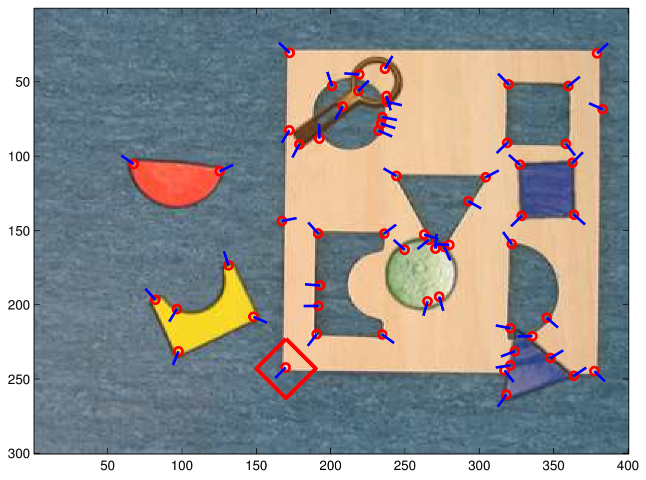


Figure 4: Directed interest points from figure 1. Square at lower centre of image is an example of a local patch.

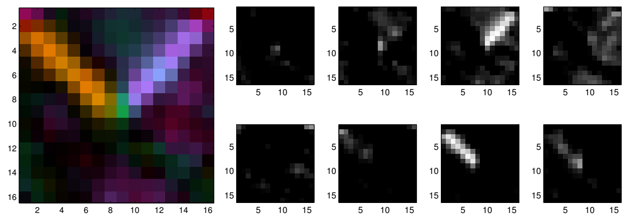


Figure 5: Local orientation patch corresponding to the patch in figure 4. Left: Local orientation patch centred in div point and oriented according to div direction. Each position in the patch is a complex value, $\mathbf{z}$. Here $|\mathbf{z}|$ is mapped to luminance, and $\arg \mathbf{z}$ is mapped to different colours. Right: The eight channels of the channel encoded patch.

The IP detector can be realised in other ways as well but some local direction at the IP is required for the later steps. The requirement of a local direction at the IP is also fulfilled by e.g. SIFT-features (Lowe 2004). For an example of directed interest points, see figure 4.

Around each IP a local orientation image patch is extracted. The patches are extracted in a local coordinate system, centred in the IP, and oriented according to the IP direction. This method results in a feature map which is invariant to rotation. See figure 5 for an example of an extracted orientation patch.

Channel Representation

The orientation of each element in the patches is *channel encoded* (Granlund 2000) giving a sparse monopolar feature vector. Channel encoding means to project the *value* of a measurement onto non-negative, compact, and smooth basis functions in the co-domain (here: orientation), see figure 6.

The projection coefficients c_k form a vector, the channel vector. The channel vectors are then weighted with the magnitudes of the orientation estimates. With a 16×16 patch

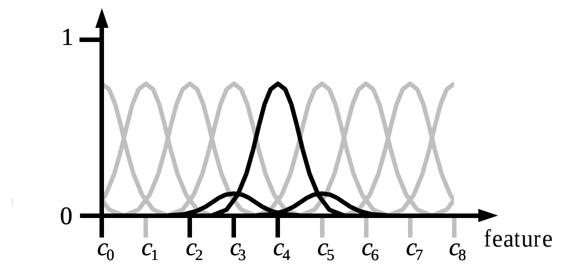


Figure 6: Channel encoding: Projection of the value c_4 onto non-negative, smooth, and compact functions (grey).

and 24 channels, this gives $16 \times 16 \times 24 = 6144$ element long *feature vector*, $\mathbf{a}$. See figure 5 (right) for an example of a channel encoded local orientation patch.

Obtaining a Response Vector

During training of the *linkage matrix* (Granlund 2000) $\mathbf{C}_1$, we also need to specify a desired response $\mathbf{u}$, see figure 3. We want $\mathbf{u}$ to describe the type, location and orientation of the object that a feature patch belongs to.

When the system enters behaviour **B3**, it has an object in the manipulator, it just does not know what it is yet. It proceeds by moving the object, and releasing it. The system then tries to recognise the object in its new location. If successful, the system leaves the object and switches to **B2**. If, on the other hand, the system fails to recognise the object, it observes what parts of the scene changed when moving the object, and generates a *change mask*. This is quite similar to the approach of Fitzpatrick (Metta & Fitzpatrick 2003), where objects are prodded in order to obtain a figure-ground segmentation. From the change mask, the system extracts an object centre, defines the object orientation as zero, and also defines a new object type. For all interest points(IP) falling inside the change mask, we define a desired response $\mathbf{u}$, from the new object type, object centre and object orientation, the latter two *in the coordinate system of the IP*.

During operation, all features (extracted around the IPs) are considered and channel encoded. It is the localised structure of the linkage matrix $\mathbf{C}_1$ which leads to an implicit segmentation of the input, depending on the respectively associated features.

Learning Algorithm

The feature vectors $\mathbf{a}$ are associated with a response vector $\mathbf{u}$, representing the object type, orientation and centre in the IP's coordinate system. In operation mode the association is formed through an associative network $\mathbf{u} = \mathbf{C}_1\mathbf{a}$. For each IP, the orientation and position are channel encoded. With 18 channels for each position coordinate, 8 channels for orientation and 6 different objects, this gives $(18+18) \times 8 \times 6 = 1728$ response channels.

The associative network is trained by a non-negative least-squares optimisation. Given a set of corresponding feature vectors $\mathbf{a}$ and response vectors $\mathbf{u}$ collected in form of column vector matrices $\mathbf{A}$ and $\mathbf{U}$, the optimisation problem

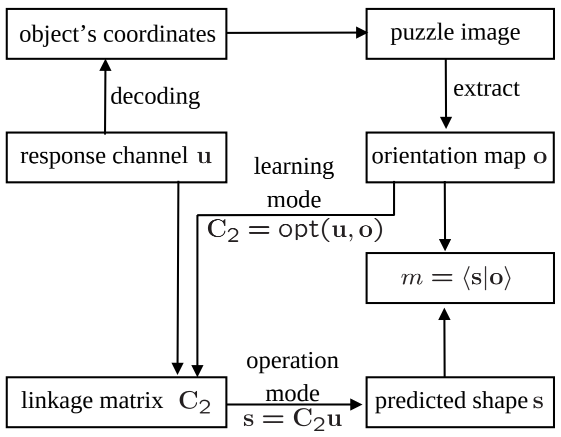

object's coordinates
puzzle image
decoding
extract
response channel $\mathbf{u}$
orientation map $\mathbf{o}$
learning mode
$\mathbf{C}_2 = \mathtt{opt}(\mathbf{u}, \mathbf{o})$
$m = \langle \mathbf{s} | \mathbf{o} \rangle$
operation mode
linkage matrix $\mathbf{C}_2$
$\mathbf{s} = \mathbf{C}_2\mathbf{u}$
predicted shape $\mathbf{s}$

Figure 7: Recognition part 2 of our system. The predicted shape is represented by channel vector $\mathbf{s}$, the local orientation map by channel vector $\mathbf{o}$. In **B2**, the associative network is replaced with a direct percept of gripping success.

reads (Johansson 2004)

$$\min_{\mathbf{C}_1 \geq 0} \|\mathbf{U} - \mathbf{C}_1 \mathbf{A}\|^2 \quad . \tag{1}$$

We will not go into further detail about the optimisation itself, the reader is referred to (Johansson 2004).

Shape Prediction and Verification

The response vectors $\mathbf{u}$ of the network $\mathbf{C}_1$ are not directly used for recognition. Instead the decoded responses are verified by predicting a channel encoded orientation patch in the object's coordinate system, using a second associative network $\mathbf{C}_2$, see figure 7.

With 6 objects and a $16 \times 16 \times 8$ orientation patch, this gives a linkage matrix $\mathbf{C}_2$, of size 2048×6. The prediction $\mathbf{s}$ is used to verify the hypothesis of object position, orientation and type, given by decodings from $\mathbf{u}$, by computing a *match score* m between the predicted orientation patch $\mathbf{o}$, and what is present in the image at the predicted position.

Voting and Hypothesis Combination

The responses $\mathbf{u}$ from the first network at each interest point are decoded and transformed to a global coordinate system, where they are again channel encoded, and weighted with the match score for the predicted shape. Since linear transformations can be performed directly in the channel representation (Felsberg, Scharr, & Forssén 2002), we symbolise the new channel vector as

$$\mathbf{v} = m\mathbf{T}(\mathbf{u}) \quad . \tag{2}$$

The resultant responses $\mathbf{v}_k$ are then summed over all IPs k. The recognition result is finally obtained as a point in the 4-D space of object type, position, and orientation, which is the channel decoding result from

$$\sum_{k=1}^{K} \mathbf{v}_k = \sum_{k=1}^{K} m_k \mathbf{T}_k(\mathbf{u}_k) = \sum_{k=1}^{K} \langle \mathbf{s}_k | \mathbf{o}_k \rangle \mathbf{T}_k(\mathbf{u}_k) \quad . \tag{3}$$

Channel Reinforcement Learning of Movements

The mapping between the control signals to the manipulator and the manipulator's movements in the image is learned using reinforcement learning (Sutton & Barto 1998). As update rule, we use the channel reinforcement technique described in (Borga 1998). Three mappings are learned: movement direction, distance, and rotation. The inputs are direction to the point to move to, distance to the point and desired rotation. The direction and distance mappings are rewarded based on the distance change to target point $\mathbf{x}^*$:

$$\Delta d = ||\mathbf{x}_0 - \mathbf{x}^*|| - ||\mathbf{x} - \mathbf{x}^*|| \, . \tag{4}$$

The rotation maping is rewarded based on the orientation error change, with respect to the target direction ϕ^*:

$$\Delta d = |\arg e^{i\phi_0 - i\phi^*}| - |\arg e^{i\phi - i\phi^*}| \, . \tag{5}$$

We define the reward for all mappings as:

$$r = 1 + \alpha |\Delta d| \Delta d \, , \tag{6}$$

where α is a tuning parameter.

Alignment

The last behaviour, **B4**, is purely rule-based and is going to be replaced with more advanced techniques in the future. The object alignment is done by trying to pick up an object on the position with highest confidence, i.e., the strongest mode of $\mathbf{v}$. If the "pick up" action succeeds, the system moves the gripped object to the position with the second highest confidence *with the same object type*. It then aligns the orientations of the objects and then releases the object in the manipulator. If the system fails to grip the object it reverses the order, i.e., it tries to grip the same object type with the weaker confidence. This is done until all objects have been moved.

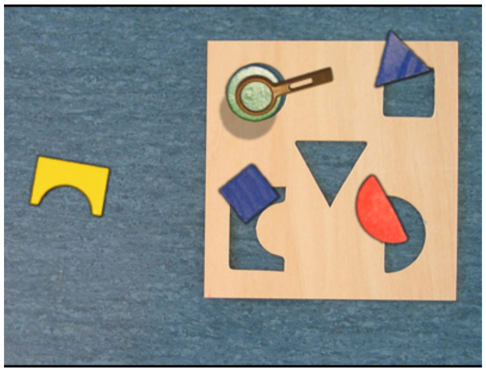
Experiments

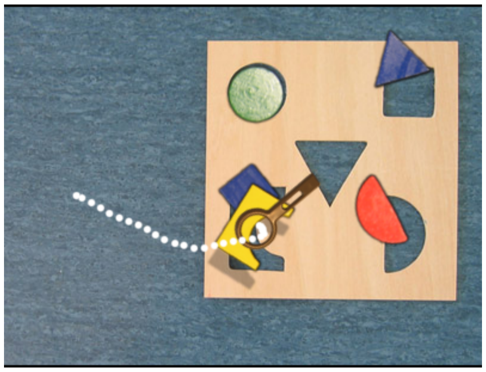
Once the system has been trained, it can be used to solve puzzle problems generated by the puzzle generator. One of these problems and its solution sequence is documented in figure 8.

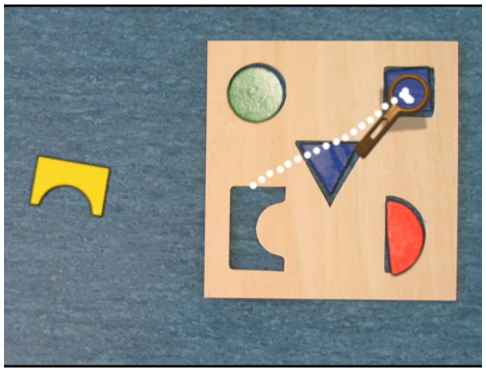
The system starts with selecting one object (the circle) and moves it to the corresponding hole, see figure 8 a. The system then selects the arch, which cannot be put into the corresponding hole, since it is occluded by the square, see figure 8 b. The system puts the arch back to its original position and continues with another object. After the triangle and the semi-circle, the system selects the square, such that the occlusion of the arch-hole is removed, see figure 8 c. Finally, the arch is put in the correct hole and the puzzle is solved, see figure 8 d.

Conclusion

It turned out that the described, simple sequence of feature associations, view based hypothesis validations, and voting lead to a robust object recognition technique which does not require a prior segmentation. The method is robust under occlusion and many changes of appearance (e.g., colour, intensity). Since our approach does not depend on particular


a: ordinary situation


b: conflict


c: solving conflict

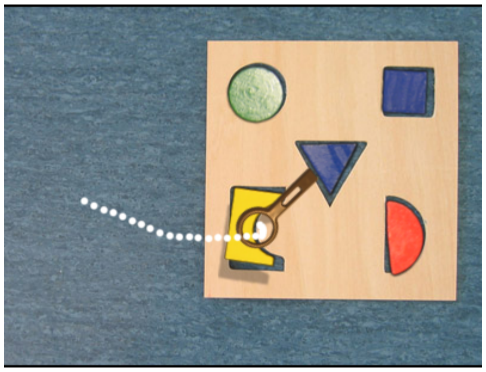

d: finished

Figure 8: Solving the puzzle.

models, we can apply it to totally different 2D settings without rewriting modules. It is just the linkage matrices which have to be relearned or extended.

For the future we hope to be able to extend the method to 3D scenarios without structural changes in the method, i.e., we replace feature spaces with higher dimensional ones, but we need not redesign the data-flow and interdependencies.

To reach the goal of a full-fledged COSPAL system the difficult steps to physical embodiment and to integration of symbolic learning and reasoning instead of hard-wired behavioral schemes have to be made. We believe that the regular APL structure of our predefined behaviors can also serve as a generating pattern for higher levels of the system and that by symbolic learning new and higher level capabilities can be generated. Without such capabilities, the system will always have to behave according to the prescribed patterns.

References

Borga, M. 1998. *Learning Multidimensional Signal Processing*. Ph.D. Dissertation, Linköping University, Sweden, SE-581 83 Linköping, Sweden. Dissertation No 531, ISBN 91-7219-202-X.

Felsberg, M.; Scharr, H.; and Forssén, P.-E. 2002. The B-spline channel representation: Channel algebra and channel based diffusion filtering. Technical Report LiTH-ISY-R-2461, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden.

Granlund, G. H. 2000. An Associative Perception-Action Structure Using a Localized Space Variant Information Representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*.

Granlund, G. 2003. Organization of Architectures for Cognitive Vision Systems. In *Proceedings of Workshop on Cognitive Vision*.

Johansson, B., and Granlund, G. 2000. Fast selective detection of rotational symmetries using normalized inhibition. In *Proceedings of the 6th European Conference on Computer Vision*, volume I, 871–887.

Johansson, B. 2004. *Low Level Operations and Learning in Computer Vision*. Ph.D. Dissertation, Linköping University, Sweden, SE-581 83 Linköping, Sweden. Dissertation No. 912, ISBN 91-85295-93-0.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2):91–110.

Metta, G., and Fitzpatrick, P. 2003. Early integration of vision and manipulation. *Adaptive Behavior* 11(2):109–128.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning, An Introduction*. Cambridge, Massachusetts: MIT Press. ISBN 0-262-19398-1.

The COSPAL consortium. 2004. Cospal project. `http://www.cospal.org`.

Vernon, D. 2004. Cognitive vision – the development of a discipline. `http://europa.eu.int/information_society/istevent/2004/cf/document.cfm?doc_id=568`.