# Combining Reinforcement Learning and Belief Revision - A Learning System for Active Vision

Thomas Leopold[1] Gabriele Kern-Isberner[1] Gabriele Peters[2]

1. University of Technology Dortmund, Germany
2. University of Applied Sciences and Arts Dortmund, Germany

thleopold@hotmail.com, gabriele.kern-isberner@cs.uni-dortmund.de, gabriele.peters@fh-dortmund.de

**Abstract**

Computer vision can highly benefit from modern learning methods. In the context of an active vision environment we introduce a machine learning approach which is able to learn strategies of object acquisition. We propose a hybrid learning method, called Sphinx, that combines two approaches originating from seperate disciplines of computer science, namely reinforcement learning on the one hand and belief revision on the other. The former represents knowledge in a numerical way, while the latter is based on symbolic logic and allows reasoning. Sphinx is designed according to human cognition and interacts with its environment by rotating objects depending on past perceptions to acquire those views which are advantageous for recognition. Our method was successfully applied in simulations of object categorization tasks.

## 1 Introduction

One of the most challenging tasks of computer vision systems is the recognition of known and unknown objects. An elegant way to achieve this is to show the system some samples of each object class and thereby train the system, so that it can recognize objects that it has not seen before, but which look similar to some objects of the training phase (due to some defined features). For this purpose several methods have been successfully used and analyzed. One of them is to set up a rule-based system and have it reason, another one is to use numerical learning methods such as reinforcement learning. Both of them have advantages, but also disadvantages. Reinforcement learning yields good results in different kinds of environments, but its training is time consuming, since it is a trial-and-error method and the agent has to learn from scratch. The possibilities to introduce background knowledge (e.g., by the choice of the initial values of the Q-table) are more limited as for example with knowledge representation techniques. Another disadvantage consists in a limited possibility to generalize experiences and thus to be able to act appropriately in unfamiliar situations. Though some generalization can be obtained by the application of function approximization techniques, the possibilities to generalize from learned rules to

BMVC 2008 doi:10.5244/C.22.48

unfamiliar situations are more diverse again with knowledge representation techniques. Knowledge representation and belief revision techniques have the advantage that the belief of the agent is represented quite clearly and allows reasoning about actions. The belief can be extended by new information, but needs to be revised when the new information contradicts the current belief. One drawback is that it is difficult to decide which parts of the belief should be given up, so that the new belief state is consistent, i.e., without inherent contradictions.

In this paper we will present our hybrid learning model Sphinx, named after the Egyptian statue of a hybrid between a human and a lion. It combines the advantages of both approaches and diminishes the disadvantages, thus synergy effect can emerge. Summarizing, the problem we address is not object recognition, rather we propose a machine learning system which interacts with its environment and is able to learn autonomously strategies to explore the environment, i.e., to learn object acquisition strategies. To demonstrate the capabilities of Sphinx we test it - in a coarse manner - in a first object classification task. Section 2 summarizes related work. In section 3 we describe our method in detail. Section 4 summarizes results from experiments carried out in object recognition environments. Finally, in section 5, we draw our conclusions.

## 2 Related Work

Psychological findings propose a two-level learning model for human learning [2], [8], [4], [11]. On the so called bottom level, humans learn *implicitly* and acquire *procedural* knowledge. They are *not aware* of the relations they have learned and can hardly put it into words. On the other level, the top level, humans learn *explicitly* and acquire *decalarative* knowledge. They are *aware* of the relations they have learned and can express it, e.g., in form of if-then rules. A special form of declarative knowledge is *episodic* knowledge. This kind of knowledge is not of general nature, but refers to *specific* events, situations or objects. Episodic knowledge facilitates to remember specific situations where general rules do not apply. These two levels do not work separately. Depending on what is learned, humans learn top-down or bottom-up [12]. It has been found [9] that in completely unfamiliar situations mainly implicit learning occurs and procedural knowledge is acquired. The declarative knowledge is formed afterwards. This indicates that the bottom-up direction plays an important role. It is also advantageous to continually verbalize what has been learned and thus speed up the acquisition of declarative knowledge. Sun, Merrill and Peterson developed the learning model CLARION [10]. It is a two-level, bottom-up learning model which uses Q-learning for the bottom level and a set of rules for the top level. The rules have the form 'Premise $\Rightarrow$ Action', where the premise can be met by the current state signal of the environment. For the maintainance of the set of rules (i.e., adding, changing and deleting rules) the authors have developed a certain technique. They have proven their model, which works similar to human learning, to be successful in a mine field navigation task. Ye et al. [13] propose a neural fuzzy system. Like CLARION, this is a two-level learning model, combining reinforcement learning and fuzzy logic. The system has successfully been applied to a mobile robot navigation task.

# 3 The Sphinx Learning Approach

Like the CLARION system, our learning approach consists of two levels. For the bottom level we use $Q(\lambda)$-Learning. For the top level we utilize belief revision techniques and an ordinal conditional function (OCF) to represent the epistemic state of an agent. Ordinal conditional functions are also called ranking functions, as they assign a degree of disbelief or surprise to each model. In this way we obtain a framework that has a well analysed, theoretical background.

First, we briefly describe how belief revision with OCFs works. Let $v_1, \ldots, v_n$ be boolean variables. Literals of $v_i$ are $v_i$ and $\overline{v_i}$ . A model is a conjunction of literals of all variables, representing possible states of the world. An OCF $\kappa$ maps each model to $\mathbb{N} = \{0;1;2;\ldots\}$. A model $\omega$ is the more plausible for smaller values of $\kappa(\omega)$. Models that are mapped to 0 are believed, i.e., the agent considers them to be most plausible to represent the true current state of the world. It might be uncertain about this current state, though, so more than one model may be mapped to 0 by the OCF $\kappa$. Moreover, $\kappa$-values can be calculated for any propositional formula $A$ by setting $\kappa(A) = min\{\kappa(\omega) \mid \omega \models A\}$. This means that a formula is considered as plausible as its most plausible models. A formula $A$ is believed iff $\kappa(A) = 0$ and $\kappa(\overline{A}) > 0$. In particular, each literal $v_i$ corresponding to one of the characterizing features is believed iff $\kappa(v_i) = 0$ and $\kappa(\overline{v_i}) > 0$. OCFs also cover the plausibility of if-then rules. An if-then rule (in this context called 'conditional') has the form $(B \mid A)$ and is similar to the classical rule $A \Rightarrow B$ but is interpreted in a three-valued context here. $(B \mid A)$ is plausible, if $\kappa(A \wedge B) < \kappa(A \wedge \overline{B})$ [5]. OCFs can be revised with new (propositional or conditional) information, i.e., the models are assigned new $\kappa$-values, so that certain conditions are met ([1], [3]).

In the following we will briefly decribe Q-learning. The setting consists of an environment and one or more agents. An agent can interact with the environment. Usually, the environment starts in a state and ends, when one terminal state is reached. This timespan is called an episode. For each action, the agent is rewarded and it aims at collecting high rewards during an episode. Episodes consist of steps. A step is the following: The agent perceives the current state of the environment via a (numerical) state signal, e.g., an ID. It looks up that action in its memory (for Q-Learning this is expressed be the Q-Table), which seems to be the best in this situation and performs it. The environment reacts on this action by changing its state. After this change, the agent is rewarded for its action and updates its Q-table.

To combine belief revision and reinforcement learning, we added a symbolic (i.e., a conjunction of literals) representation of the states of the environment, thus states s have a dual representation $s = (s_{num}; s_{sym})$ : a numerical one for the reinforcement learning part and a symbolic one for the belief revision part. The Sphinx system is displayed in figure 1 and works as follows:

**Algorithm 'Sphinx-Learning':**
1. The Sphinx agent perceives the signal of the dually represented state $s$ coming from the environment.
2. The agent queries its OCF $\kappa$ about which actions $A_\kappa(s) = \{a_1, \ldots, a_k\}$ are most plausible in s.
3. The agent looks up the Q-values of these actions and determines the set $A_{best}(s) \subseteq A_\kappa(s)$ of those actions in $A_\kappa(s)$ that have the greatest Q-value. (An ordinary Q-agent determines

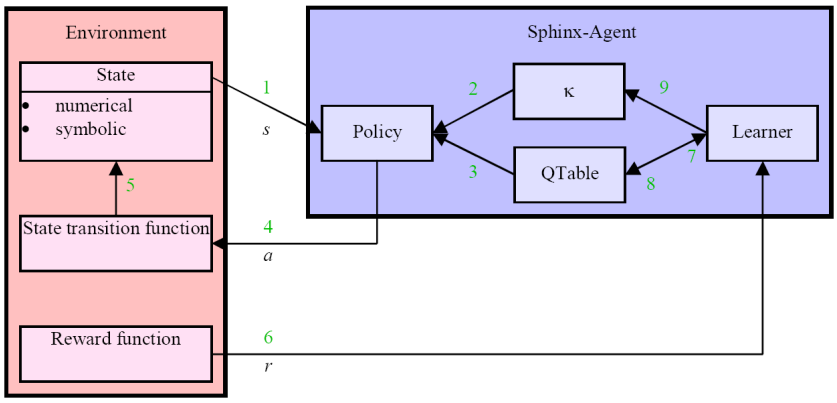


Figure 1: Sphinx Learning Approach

the set of best actions from the set of all possible actions.)
4. The agent chooses a random action a from $A_{best}(s)$ and performs it.
5. The environment changes to the successor state.
6. The agent receives the reward $r$.
7. The agent updates the Q-table.
8. The new Q-values for actions in $s$ are being read.
9. The agent revises $\kappa$. (This revision usually makes those actions most plausible in $s$ that have the greatest Q-value in $s$. In addition, the agent tries to find patterns in the state signals for which certain actions are generally better than others. If such a relation between a pattern and a set of actions is found, a revision with a generalized rule is performed which is described below.) If the current state is not a terminal state, go to 1.

For the symbolic representation of the states, i.e., the description of the states of the environment in words, suitable (finite) domains $d_i = \{v_{i.1}; \ldots; v_{i.m_i}\}$ and analogously called variables $d_i = v_{i.j}$ are defined. The symbolic representation of a specific state is the conjunction of the corresponding literals of all of these variables: $s_{sym} = (d_1 = v_{1.k_1} \wedge \ldots \wedge d_n = v_{n.k_n})$.

Furthermore, we define a domain 'Action' containing the set of possible actions. The models for $\kappa$ have the form $\underbrace{d_1 = v_{1.k_1} \wedge \ldots \wedge d_n = v_{n.k_n}}_{s_{sym}} \wedge Action = a_i$, where the first part is a symbolic representation of a state $s$ and the second part is a symbolic representaion of an action $a$ with the meaning, that if such a model is plausible due to $\kappa$, then $a$ is plausible in $s$. The set of all models for $\kappa$ is created by combining each possible state of the environment with each possible action. When a state $s = (s_{num}; s_{sym})$ is perceived (step 1 in algorithm 'Sphinx-Learning'), then $\kappa$ is searched for the most plausible models containing $s_{sym}$, i.e., for those models (containing $s_{sym}$) with the smallest value of $\kappa$. These models also contain an action. $A_\kappa(s)$ (step 2) is created by these actions. Then the actions in $A_\kappa(s)$ are filtered by their Q-values (step 3) and one of the remaining actions is carried out (step 4). Steps 5 to 7 are pure Q-learning. In step 8 the best actions for $s$ due to the new Q-values are determined. The revision of $\kappa$ with this information is a little complex

and described in the following. We revise $\kappa$ with up to four different kinds of information after each step:
1. Revision with information on a poor action in a specific state (episodic knowledge).
2. Revision with information on a poor action in several, similar states (generalization).
3. Revision with information on best actions in a specific state (episodic knowledge).
4. Revision with information on best actions in several, similar states (generalization).

A 'poor' action in a specific state or in several, similar states was defined as an action that yields a reward less than -1. The conditionals used to revise $\kappa$ are as follows:

1. $(\overline{Action = a}|s_{sym})$, where $s_{sym}$ is the symbolic representation of a certain state $s$ in which $a$ is poor.
2. $(\overline{Action = a}|p)$, where $p$ is a pattern, i.e., a conjunction of *some* of the variables describing the state of the environment. $p$ represents a set of states, which are similar, because they share a common pattern.
3. $\left(\bigvee_i Action = a_i|s_{sym}\right)$, where all $a_i$ are best actions (due to their Q-values) in $s$.
4. $\left(\bigvee_i Action = a_i|p\right)$, where each $a_i$ is a best action in at least *one* of the states covered by the pattern $p$. $a_i$ needs not to be a best action in *all* states covered by $p$.

The last form of revision has the purpose to prevent that actions, which are not best actions, are classified plausible when $p$ is perceived. So the agent has to find the best action for a specific state covered by $p$ only among the actions $a_i$. Since revisions and especially revisions with generalized rules have a strong influence on the choice of actions, they have to be handled carefully, i.e., the agent should be quite sure about the correctness of a rule before adding it to its belief. Therefore, the agent uses several counters counting, how 'often' an action has been poor, not poor, a best or not a best one under certain circumstances. With these counters probabilities can be calculated which can be used to evaluate the certainty about the correctness of a specific rule. Our learning model also supports background knowledge. If the user knows some rules that might be helpful for the agent and its task, she can formulate them as conditionals and let the agent revise $\kappa$ with them before starting to learn.

# 4 Results

We tested the Sphinx approach in a navigation environment and in two different simulations in a viewpoint planning context. In this paper, we present the results of the latter.

## 4.1 Recognition of Geometric Objects

In this test environment, the agent has to learn to recognize the following objects: sphere, ellipsoid, cylinder, cone, tetrahedron, pyramid, prism, cube, and cuboid. By interacting with the environment the agent can look at the object from the front, from the side or from the top or it can choose to try to name the current object.

| Number of Appearances per Object | Recognition Rate (in %) ($Q(\lambda)$) for $\lambda = 0.5$ | Recognition Rate (in %) (Sphinx without background knowledge) | Recognition Rate (in %) (Sphinx with background knowledge) |
|---|---|---|---|
| 10 | 27.5 | 29.7 | 35.4 |
| 20 | 47.9 | 55.9 | 66.0 |
| 30 | 66.0 | 76.5 | 84.3 |
| 40 | 78.6 | 88.5 | 92.1 |
| 50 | 85.4 | 93.8 | 95.6 |
| 60 | 90.3 | 96.4 | 97.4 |
| 70 | 92.5 | 97.7 | 98.5 |
| 80 | 94.4 | 98.5 | 99.2 |
| 90 | 95.5 | 98.9 | 99.4 |
| 100 | 96.4 | 99.4 | 99.5 |

Table 1: Recognition Rates for Geometric Objects

The possible front, side, and top views are represented by five elementary shapes, namely: circle, ellipse, triangle, square, and rectangle. For example, the cone has the front view 'triangle', the side view 'triangle', and the top view 'circle'. This leads to the following domains for this environment:

- $\mathit{FrontView} = \{\mathit{Unknown}, \mathit{Circle}, \mathit{Ellipse}, \mathit{Triangle}, \mathit{Square}, \mathit{Rectangle}\}$
- $\mathit{SideView} = \{\mathit{Unknown}, \mathit{Circle}, \mathit{Ellipse}, \mathit{Triangle}, \mathit{Square}, \mathit{Rectangle}\}$
- $\mathit{TopView} = \{\mathit{Unknown}, \mathit{Circle}, \mathit{Ellipse}, \mathit{Triangle}, \mathit{Square}, \mathit{Rectangle}\}$
- $\mathit{Action} = \{\mathit{LookAtFront}, \mathit{LookAtSide}, \mathit{LookAtTop}, \mathit{RecognizeUnknown}, \mathit{RecognizeSphere}, \mathit{RecognizeEllipsoid}, \mathit{RecognizeCylinder}, \mathit{RecognizeCone}, \mathit{RecognizeTetrahedron}, \mathit{RecognizePyramid}, \mathit{RecognizePrism}, \mathit{RecognizeCube}, \mathit{RecognizeCuboid}\}$

At the beginning of each episode, the environment chooses one of the nine geometric objects and generates the state signal '$\mathit{FrontView} = \mathit{Unknown} \wedge \mathit{SideView} = \mathit{Unknown} \wedge \mathit{TopView} = \mathit{Unknown}$'. If the agent's action is *LookAtFront, LookAtSide,* or *LookAtTop,* then the *FrontView, SideView,* or *TopView,* respectively, is revealed in the new state signal following the agent's action. If the agent's action is an action of type '*Recognize...*' action, the episode ends. The reward function returns -1, if one of the '*Look...*' actions has been performed. Otherwise, the agent is rewarded with 10, if it has recognized the objects correctly, and with -10, if not. After ten steps the current episode is forced to end. Table 1 shows the recognition rates after each training phase. In each training phase, each object is shown ten times to the current agent. The values are averaged from 1000 independent agents. If the agent is provided with the background knowledge "If no view has been perceived yet, then look at the front, the side, or the top of the object" via the conditional $(\mathit{Action} = \mathit{LookAtFront} \vee \mathit{Action} = \mathit{LookAtSide} \vee \mathit{Action} = \mathit{LookAtTop} \,|\, \mathit{FrontView} =$

$Unknown \wedge SideView = Unknown \wedge TopView = Unknown$), the recognition rates presented in the last column of 1 are obtained. Some of the learned rules are:

- If $FrontView = Circle$, then $Action = RecognizeSphere$
- If $FrontView = Unknown \wedge SideView = Triangle$, then $Action = LookAtFront$
- If $FrontView = Triangle \wedge SideView = Unknown$, then $\overline{Action = RecognizePrism}$ .

## 4.2 Recognition of Simulated Real Objects

To analyse Sphinx under more realistic conditions, we set up another environment. We defined shape attributes that are suitable for representing objects in a simple object recognition task and then chose arbitrary objects and described them with these previously defined attributes. These attributes are the input to Sphinx. (What remains to be done at this point to apply our approach in a real active vision environment, is the extraction of these shape attributes from the images. This can be done by existing segmentation methods. Of course, this can be difficult in some cases, but it is not the subject of this contribution.) Again, there are three possible perspectives: the front view, the side view, and a view from a position between these two views. The decision for these persepectives, especially for the intermediate view, was made based on the results found by [7] who revealed that the intermediate view plays a special role in human object recognition. The front and the side view are described by three attributes each: approximate (idealized) shape, size (i.e., proportion) of the shape, and deviation from the idealized shape. The approximate shape can take the values *unknown, circle, square, triangle up,* and *triangle down*. The size can be *unknown, flat, regular,* or *tall*. The deviation can be *small, medium,* or *large*. Besides these attributes the object is described by the complexity of its texture. This attribute can take the values *simple, medium,* and *complex*. We set the attributes for each object manually. In a real viewpoint planning task they can be determined easily by a simple image processing module which merely has to quantize the shape and texture of an object. If the agent looks at the object from the front or the side, it perceives the matching idealized shape, its size, its deviation, and the complexity of the texture. From the intermediate view the agent can only perceive the idealized shapes of the front and the side view and the complexity of the texture, but not the size and deviations. Formally the domains are:

- $FrontViewShape = \{Unknown, Circle, Square, TriangleUp, TriangleDown\}$
- $FrontViewSize = \{Unknown, Flat, Regular, Tall\}$
- $FrontViewDeviation = \{Unknown, Small, Medium, Large\}$
- $SideViewShape = \{Unknown, Circle, Square, TriangleUp, TriangleDown\}$
- $SideViewSize = \{Unknown, Flat, Regular, Tall\}$
- $SideViewDeviation = \{Unknown, Small, Medium, Large\}$
- $Texture = \{Simple, Medium, Complex\}$
- $Action = \{RotateLeft, RotateRight, RecognizeUnkown\} \cup R$, where $R$ is the set of '*Recognize...*' actions.

Figure 2: Two Examples from the Object Classes Bottle and House. The two left hand images show the front and side view of a bottle with medium texture. Shape, size, and deviation for both views are triangle up, tall, and medium, respectively. These attributes are the input for 'Sphinx'. The two right hand images show the front and side view of a house with medium texture, as well. Shape, size, and deviation of the front view are square, regular, and little, of the side view triangle up, regular, and medium.

At the beginning of each episode, the agent looks at the current object from a random perspective and the variables are set according to this perspective. Now the agent can rotate the object clockwise or counter-clockwise or name it. If the agent's action is a '*Recognize...*' action, the episode ends. After ten steps the running episode is forced to end. The reward function is the same as in the previous test environment. We have chosen 15 different objects from nine different object classes such as bottle, tree, and house for which we provide the three attributes mentioned (shape, size, and deviation) (figure 2). Table 2 shows the results averaged from 100 independed agents.

In a second step we added background knowledge that enables the agent to recognize all objects correctly, if it has perceived all of the three views. Furthermore, we added rules to the background knowledge that told the agent to look at the object from all perspectives first. With these rules the agent has a complete, but not optimal, solution for the task. We wanted to find out how fast the agent learns that it does not need all views to classify the current object. To protect the background knowledge from being overwritten by the agent's own rules too early, some parameters were changed, so that the agent had to be more sure about the correctness of a rule before adding it to its belief. This setup resulted in a constantly high recognition rate from 99.3% to 99.8%. The number of perceived views decreased over time from 3.28 to 1.99. (The value of 3.28 perceived view vs. 3 possible views results from the fact, that the intermediate view has to be perceived twice if the environment starts in this view. Then, the agent perceives this view at the beginning, then rotates the object to the front and then back to the intermediate view so it can rotate the object to the side view in the next step, or vice versa.)

Here are some of the rules the agent learned and assimilated during its training:

- If $FrontViewShape = TriangleUp \wedge FrontViewSize = Tall$, then $Action = RecognizeBottle$
- If $FrontViewShape = Circle \wedge SideViewShape = Unknown \wedge Texture = Simple$, then $Action = RotateLeft$
- If $Texture = Complex$, then $\overline{Action = RecognizeBottle}$

| Number of Appearances per Object | Recognition Rate (in %) ($Q(\lambda)$) for $\lambda = 0.5$ | Recognition Rate (in %) (Sphinx without background knowledge) | Recognition Rate (in %) (Sphinx with background knowledge) |
|---|---|---|---|
| 10 | 40.1 | 46.3 | 99.6 |
| 20 | 63.1 | 70.7 | 99.3 |
| 30 | 78.2 | 86.0 | 99.4 |
| 40 | 86.3 | 91.6 | 99.5 |
| 50 | 89.6 | 94.7 | 99.4 |
| 60 | 91.1 | 95.7 | 99.7 |
| 70 | 92.2 | 97.3 | 99.7 |
| 80 | 93.0 | 98.0 | 99.7 |
| 90 | 93.8 | 98.4 | 99.7 |
| 100 | 94.5 | 98.8 | 99.8 |

Table 2: Recognition Rates for Simulated Real Objects

These rules can be comprehended best with the complete set of used objects at hand. They are depicted in [6].

# 5 Conclusion

We have proposed a new machine learning approach for active vision that is able to learn object acquisition strategies autonomously. To evaluate the system we applied it in a simulated object recognition task. This simulation is not based on real images, rather we provide the system with simple shape descriptors which are the input to our learning system. These shape attributes can be determined from real images by standard image processing techniques, which is not subject of this contribution. Of course, in future research we will test our system also in real world object recognition tasks. In more detail, the contributions of this paper are as follows. We have proposed the hybrid learning method Sphinx and applied it in a simulation of an interactive object recognition task. Sphinx has an advantage over classic reinforcement learning in terms of learning speed by combining two cognitive levels of learning - similar to human learning. This advantage remains even if Sphinx is not provided with background knowledge., i.e., when both learning methods are started equivalently.

Nevertheless, background knowledge can support the process of learning, so that even at the beginning high success rates can be achieved. It can be described in a manner comprehensible for humans in the form of conditionals. This is not feasible in classical reinforcement learning. Moreover, learned information can be output easily comprehensible for humans in the form of rules; there is no need to interprete numerical data. By providing Sphinx with background knowledge in the form of obvious (but not necessarily optimal) rules permanently high learning rates can be obtained, while the amount of required information to choose a proper action decreases. These characteristics qualify our

hybrid learning method especially to be applied in active vision environments.

## Acknowledgments.

This research was funded by the German Research Association (DFG) under Grant PE 887/3-2.

## References

[1] Alchourrón, C. and Gärdenfors, P. and Makinson, D. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[2] Anderson, J. R. *The architecture of cognition*. Hardvard University Press, Cambridge, MA, 1983.

[3] Darwiche, A. and Pearl, J. On the logic of iterated belief revision. *Artificial Intelligence*, 89(1-2):1–29, 1997.

[4] Gombert, J.-E. Implicit and explicit learning to read: Implication as for subtypes of dyslexia. *Current Psychology Letters*, 10(1), 2003.

[5] G. Kern-Isberner. *Conditionals in nonmonotonic reasoning and belief revision*. Springer, Lecture Notes in Artificial Intelligence LNAI 2087, 2001.

[6] Leopold, Thomas. Reinforcement Learning unter Benutzung von Hintergrundwissen und Revisionstechniken mit Anwendung auf interaktive Objekterkennung, 2007.

[7] Pereira, A. and James, K. H. and Jones, S. S., and Smith, L. B. Preferred views in children's active exploration of objects, 2006.

[8] Reber, A. S. Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*, 118(3):219–235, 1989.

[9] Stanley, W. B. and Mathews, R. C. and , Buss, R. R. and Kotler-Cope, S. Insight without awareness: On the interaction of verbalization, instruction and practice in a simulated process control task. *The Quarterly Journal of Experimental Psychology Section A*, 41(3):553–577, 1989.

[10] Sun, R. and Merrill, E. and Peterson, T. From implicit skills to explicit knowledge: a bottom-up model of skill learning. *Cognitive Science*, 25(2):203–244, 2001.

[11] Sun, R. and Slusarz, P. and Terry, C. The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review*, 112(1):159–192, 2005.

[12] Sun, R. and Zhang, X. and Slusarz, P. and Mathews, R. The interaction of implicit learning, explicit hypothesis testing learning and implicit-to-explicit knowledge extraction. *Neural Networks*, 20(1):34–47, 2007.

[13] Ye, C. and Yung, N. H. C. and Wang, D. A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(1):17–27, 2003.