Project n:
FP6 - 511 931

Project acronym:
MIND RACES

Project title:
MIND RACES: from Reactive to Anticipatory Cognitive Embodied Systems

Deliverable D14 (D4.2): Experimental results and benchmarking of mechanisms based on analogy, proactive and goal directed behavior.

Period covered from/to: 01.06.2005 – 05.04.2007 Date of preparation: 17/04/2007

Start date of the project: 01.10.2004 Duration: 36 months

Organization Name of Lead Contractor For This Deliverable: Revision: 01
UW (Department of Cognitive Psychology, University of Würzburg)

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	**Public**	
PP	Restricted to other programmes participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	X

Document identifier:	**DELIVERABLE_D4.2(D14)**
Date:	**17/04/2007**
Work package contributing to the deliverable:	**WP 4: Goal-directed behavior, Pro-activity and Analogy**
Partner(s):	**UW, LUCS, IDSIA, OFAI, ISTC-CNR, NBU**
Lead Partner:	**UW**
Document status:	**draft**
Deliverable identifier:	**WP4_D4.2**

Delivery Slip

	Name	**Partner**	**Date**	**Signature**
Author	Martin Butz	UW	13.04.2007	
Verified	Rino Falcone	ISTC_CNR	16_04_2007	
Approved by	Rino Falcone	ISTC_CNR	16_04_2007	

Project information

Project acronym:	Mind Races
Project full title:	MIND RACES: from Reactive to Anticipatory Cognitive Embodied Systems
Proposal/Contract no.:	IST-511931
Project Manager:	ISTC_CNR
WP Manager:	Martin V. Butz
Address:	Department of Cognitive Psychology III Röntgenring 11 97070 Würzburg Germany
Phone:	+49 931 31 2808
Fax:	+49 931 31 2815
Mobile: E-mail	+49 172 971 9769 mbutz@psychologie.uni-wuerzburg.de

TABLE OF CONTENTS

PART 2 – Management Overview

Document Control

This document is a co-production of all the partners mentioned above. First, contributions from all the partners were gathered (initiated on 12.03.2007, but already mentioned at the meeting in October and December 2006 and also by Rino Falcone in February 2007). This document includes all descriptions received until 05.04.2007. The responsible author put the information together and provided an overarching perspective of the achievements within the workpackage. The first draft version was sent to all authors for verification (11.04.2007). The final version was presented to the program coordinator on 13.04.2007.

Executive Summary

This document provides an overview over all developed systems, implementations, simulations, and evaluations relevant to Workpackage 4: Goal-directed behavior, pro-activity, and analogy. The workpackage is mainly concerned with anticipatory (goal-directed) action decision making and action control that is controlled by predictions and analogical reasoning. Thus, work is included that assesses the benefits of anticipatory processing mechanisms, classifies anticipatory mechanisms for behavioral control, or applies anticipatory behavioral control mechanisms in simulated or real robot environments. The results provide a broad spectrum of successfully accomplished system enhancements of anticipatory processes in different cognitive architectures. Additionally, advanced conceptualizations of anticipatory processes and anticipatory behavior were developed. This deliverable gives an overview over the accomplishments in the MindRACES project. However, also work in progress is reported, which, due its partially highly challenging character, may be continued beyond the boundaries of the project, which is scheduled to end at the 30th of September 2007.

While the previous deliverable in workpackage 4 (Del. 4.1, i.e., Del 5) showed the commonalities of various systems the MindRACES partners were interested in, this workpackage points out successful system combination due to successful collaborations and comparative studies between various MindRACES partners. Many suggested potential collaborations and system combinations from deliverable 4.1 were successfully carried through and were being published or are close to submission or publication. The general challenge to combine the various systems, investigate their anticipatory capabilities, and consequently develop more anticipatory cognitive embodied systems was met in several respects as outlined in this deliverable.

PART 3 – Deliverable Content

1 Introduction

While deliverable 4.1 (#5) reported multiple facets of predictive and anticipatory systems, this deliverable focuses on the conceptual and system advancements achieved by the partners over the last two years. We focus on published or submitted articles but also mention work that is still in progress. Before moving on to the individual contributions, we provide a short overview over the different facets of anticipatory behavior for goal-directed behavior, pro-activity, and analogy making.

In deliverable 4.1, we distinguished various anticipatory capabilities of the project-relevant systems. These systems included (1) the DYNA architecture and related systems such as the XCS/XACS, NN-based DYNA models, and the artificial immune system architecture (AIS). (2) Inverse model-based systems as well as the inverse gradient method that can exhibit goal-initiated behavior, that is, behavior that is triggered by the activation of a current (reachable) goal. In the case of inverse gradient methods, the behavior may also be combined with reinforcement learning capabilities. (3) Context-based systems provide a useful tool of how to include context information to guide and direct anticipatory mechanisms not only for attention processes but also for consequent action decision making and action control mechanisms. (4) Analogy-based systems, such as the AMBR architecture, are useful to study analogy making in relation to anticipatory behavior. (5) Recurrent neural network approaches, such as LSTM and others, finally were relevant to learn efficient dynamic predictions and classifications .However, their impact on anticipatory behavior remained unclear.

Deliverable 4.1 pointed out that besides model learning improvements, several anticipatory behavior capabilities required further research effort. These included the further development of inverse modeling capabilities. Hereby, it was expected that adaptive robotic applications will be of interest and that inverse gradient approaches could help to further improve learning of such inverse models. Second, we identified behavioral adjustments due to unexpected sensory inputs as a target for further research effort. Third, directed, task-dependent planning mechanisms were mentioned as an additional beneficial target of anticipatory behavior research effort. Fourth, the coupling of motivational mechanisms with behavioral decision making and control poses additional challenges. Fifth, while curious behavior had been implemented before, successful epistemic actions had not been shown in any of the relevant architectures gathered in the previous deliverable.

This short overview indicates that there remained lots of work concerning anticipatory behavior with respect to goal-directed behavior. We now present the advancements made over the last two years that address all of these issues. Albeit certainly never done and over, all the efforts provide some bits and pieces of how anticipatory behavior can be improved with respect to the five types of system approaches and with respect to the behavioral aspects summarized above.

2 Advancements of Predictions and Anticipations

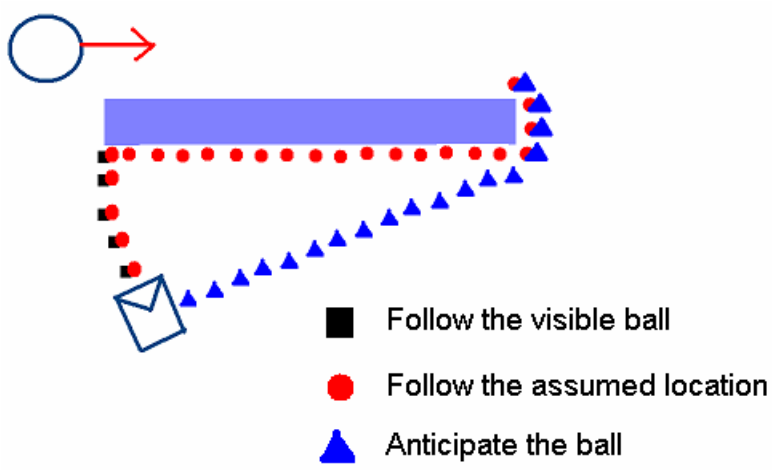
2.1 Predictive Particles: A Novel Approach for Efficient Predictions In Non-Markov Environments (OFAI)

2.1.1 Idea

To develop more predictive capabilities and model fuzzy predictions in an environment with obstacles, the Predictive Particles (PP) approach assumes that a robot and a moving target act in a known environment. Obstacles, predominantly walls, may hinder free view and motion. The task of the robot is to anticipate a possible location of reappearance, if the target is currently hidden. The belief about the current state of the target can be expressed as a time-dependent probability function, which itself is estimated by a sample of particles.

So far, it was assumed that the robot has a complete map of the environment. The robot uses its own observations to localize itself and the target. All its observations of the target are transformed into

the world view and are used to establish a motion model within the world view. It is assumed that, given the map of the environment, the motion dynamics of the target are fully described by its current location and its velocity vector. In particular, the target does not take notice of the robot and will not adapt its behavior depending on actions of the robot. Furthermore the behaviour of the target will always depend only on its current movement within its immediate neighborhood.



Particle filters are becoming more and more popular in the robotics community. Thrun et. al (p.437, 2000) mention "that particle filters are at the core of some of the most effective robotics algorithms". In chapter 13, particle filters are used to treat the SLAM (Simultaneous Localization and Mapping) problem, but here, in our approach, the robot needs to localize itself. Approaches using particle filters for target tracking have already been proposed: Schulz et al. developed a hybrid version using particle filters to simultaneously predict the trajectories of multiple (but visible) objects.

The approach of Mottaghi et al. (2006) is similar to one pursued. They assume that an intruder should be tracked down. The current position of the intruder is approximated by a sample of particles. Mottaghi et al. (2006) put the emphasis on the cooperation of several robots. Their dynamic model to extrapolate particle positions differs from our model mainly in the point that the model from Mottaghi et al. (2006) always uses the *current* position of particles, whereas our approach also use *forecasted* positions. Therefore, their model differs mainly in the way, how the direction to which the robot(s) are supposed to proceed is generated.

2.1.2 Particle filters

Applying a particle filter is an alternative nonparametric implementation of the Bayes filter (for an elaborate description see e.g. Thrun et al., 2000). The object of interest is the current belief about the state of the target, expressed by a probability distribution over location and velocity. As time passes by, the probability distribution of the state of an unobserved target becomes more and more complicated. The presence of obstacles generates multimodal distributions, as the target might have taken either of both ways to pass the obstacle. Particle filters approximate this complicated posterior distribution of the state by a sample of particles distributed according to this distribution.

In the case of a visible target, particles whose locations are nearer to the observed location, get higher weights. Strictly speaking, we should have used also the similarity between particle velocity and measured velocity to calculate the weights, as a particle with the correct location but wrong velocity will still get a high weight. If we assume that the target can be observed for 2 or more time steps, such a particle will vanish with high probability, and it is usually safe to work with only the location. If the target is invisible, those particles which are “visible” get a low weight, which might be even zero, and all “hidden” particles get a higher weight, as they do not contradict the observation. The cloud of particles tends to be relatively narrow (depending on the quality of sensors), if the target is visible, and starts to grow for an invisible target.

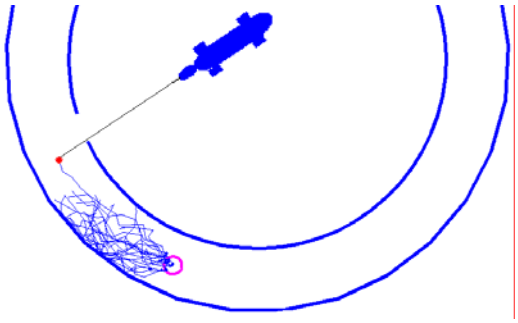
2.1.3 Motion of the robot

The approach assumes that the main task for the robot should not be the hunting of a visible target, but to anticipate the next appearance, assuming that walls often do not allow an unobstructed view of the target. The decision, whether the robot will be able to catch the target at the forecasted position depends on the distance from the robot to this point and on the forecasted time span the target might need to reappear. As it is assumed that the robot can estimate the velocity of the target and knows its own velocity, it can calculate the time needed to reach this point and also, whether the robot is sufficiently fast. In the case that the time is too short, the robot could decide to forecast even further and wait for the next attempt to catch the target. As the amount of calculations grows linearly with the forecast horizon, this approach is not pursued, though. Rather, it is assumed that the robot is sufficiently fast. Thus, sending him to the forecasted location of reappearance will usually at least decrease the distance to the target.

In sum, the robot "thinks" only in particles. Actually observing the target gives the possibility to improve the current quality of the particle representation of the true state. Any intended motion just relies on the particle locations. ´Thus, a reactive behavior is assumed in the sense that the robot will try to follow particles if they are visible, which means that the robot does not view any obstacles between his and the particle position.

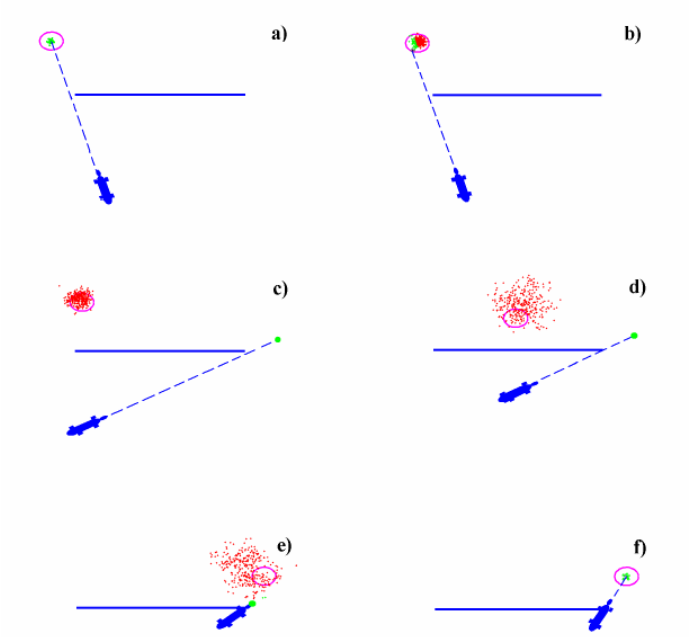
If the robot follows any visible particle and the target does not appear, then the visible particles will most likely vanish after a short time, as their weights are 0 or at least lower than the weights of hidden particles. In this case there will be no visible particle to follow, but we can still work with the hidden particles. The robot has an internal representation of all particles that allows him to simulate possible trajectories. In this internal representation it also could "see" particles which are in reality behind his back.

The robot tries now to follow the particles which, in his internal representation, are immediately visible. The robot is turning and moves to available particles in his back. But after a short time, even those virtual particles, which, after turning, transform into "real" visible particles might disappear, if the target couldn't be found. We assume now that the robot starts to extrapolate and simulates future trajectories of all particles. It could follow different guidelines and could stop this procedure, if e.g. at least one of the extrapolated hidden particles re-appears or all of the particles re-appear.



2.1.4 Experiments in simulated worlds

The main simulation effort focused on an artificial world with a ball moving behind a single wall. The ball physics were chosen according to the law "angle of reflection equals angle of incidence", the accuracy of the vision system was assumed to be high, which is equivalent to rapidly decreasing weights by increasing distances between particles and target. The next figure illustrates the different phases which occur.



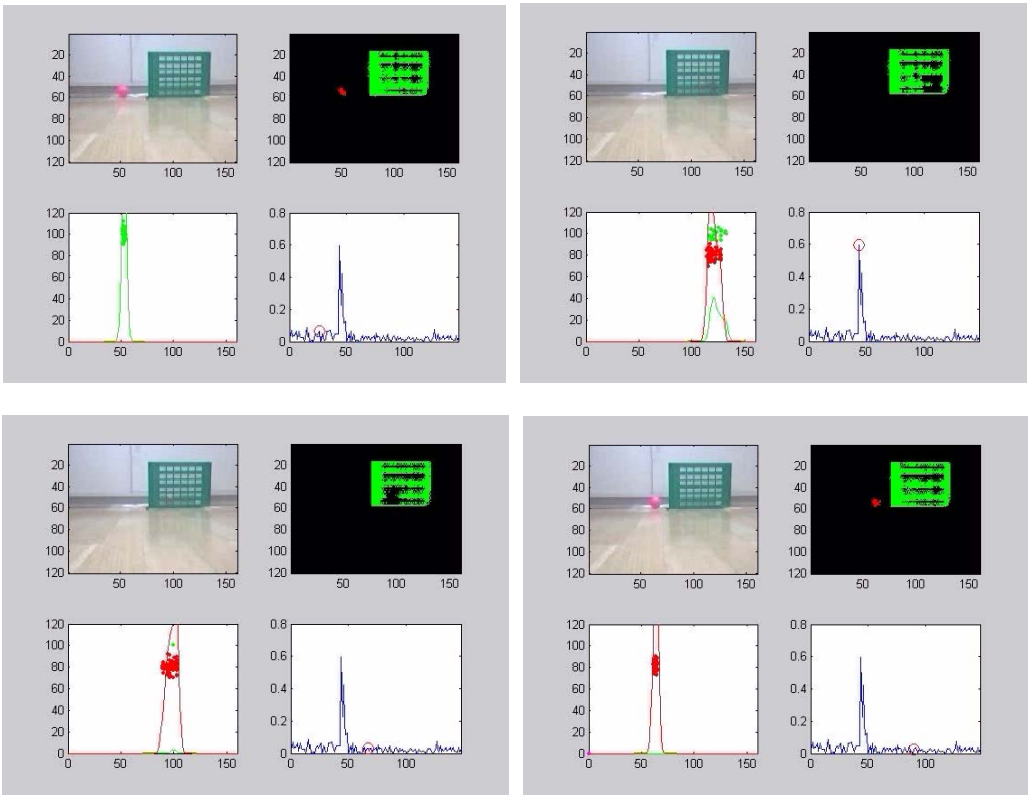
As long as the robot can observe the ball, the particle cloud tends to be small (a). If the ball vanishes, there might be some visible particles to which the robot could walk to, but they disappear within a few steps (b). If all particles are invisible (c), the robot moves immediately to the nearest visible *extrapolated* particle, which from now on happens to appear at the right end. The size of the cloud grows (d). When the first particles become visible at the right end, the robot moves to the nearest of them. As long as the ball is not visible, these particles tend to disappear (e). Finally, the ball reappears, and the particle cloud shrinks again (f).

2.1.5 Ongoing: Experiments in the real world

Besides various advanced simulations (not mentioned here), experiments with a single wall on the AIBO robot were performed. In this setting another obstacle was placed sometimes (~20 percent of all cases) behind the right edge of the wall. Hitting the obstacle produces a noise which is almost always louder than ambient noise. An additional system with two thresholds was used to decide whether an obstacle has been hit. When the amplitude of the noise signal was lower than the lower threshold, the probability of a particle of having reversed its direction is 0, if the amplitude is higher than the upper threshold the probability was set to1. In between the two thresholds, the probability was linearly interpolated. Usually it was safe to set both thresholds to the same value.

Three problems arose: AIBOs camera only recorded 15 frames per second, and additionally some of the frames were identical, sometimes 4 frames in a row, and estimation of velocities was not straightforward. Sound was delayed up to a third of a second and we were not sure, whether the delay was always the same. For the mentioned experiments, we replaced AIBO by a stationary camera with 30 frames per second and took then every 4^{th} frame. The sound delay was neglible.

The third problem was that the noise when hitting the obstacle lasts longer than one frame. As a solution we introduced an interval after the first appearance of a high amplitude (higher than the threshold) for which the acoustic signal was ignored.



The four figures show in the upper left the observed scene, the extracted objects in the upper right, the current estimate of the ball position (bottom left) and the sound during the whole run with the red circle indicating the current time (bottom right).

The ball position estimate is green for all particles going to the right and red for the opposite direction. A kernel was used to construct a density for both cases. When the ball hits the obstacle in the second picture, the direction of nearly all particles is reversed. The ball returns and is finally visible again. If the ball is visible, the width of the density is much smaller (last picture).

2.1.6 Summary and possible extensions

The algorithms were designed to combine anticipatory behavior with reactive behavior. Therefore, the robot only finds targets that are or will be visible from the current viewpoint (or from viewpoints which might be generated during the motion of the robot). Equipped with this algorithm, the robot does not have the capacity to find a target with known location that stopped behind a wall. The robot *might* find the target nonetheless in the case that particles will reach the ends of the wall often enough to draw the robot to one of both ends, from which it might see the target itself.

So far, non-decreasing velocity models were considered. If absolute velocity was allowed to decrease, e.g. due to friction, until it reaches zero, then the particles will come eventually to a stop. If the friction is so high that not a single hidden particle re-appears, the current algorithm needs to be further enhanced such that the robot was able to reach any given coordinate, visible or hidden. During the extrapolation part, for each forecast horizon a location coordinate is calculated from the simulated particle trajectories. With the help of an additional algorithm, a path may be constructed and the length of this path may be calculated. If the robot is sufficiently fast to reach the location within the given time, the direction along the path is chosen. Of course, even more sophisticated algorithms can be developed, e.g. algorithms that maximize the probability to catch the target. The simulated particle trajectories could be helpful for this task, as they might be used to estimate probabilities of having a distance lesser than a given distance to the true position of the target.

2.2 Advaced Predictive Capabilities in the XCS Classifier System (UW)

A more theoretical account of system predictability was achieved by the advancement of the function approximation capabilities of the XCS system (Butz, Lanzi, & Wilson, 2006, in press). We showed that XCS's performance can be improved in three ways. (1) Faster and more accurate linear approximation with efficient RLS stabilized and improved performance. (2) The representation of the classifier condition improved function approximation, in this case preventing unsuitable classifier overlaps. (3) The operators in the evolutionary process were optimized to enable faster learning by a more directed evolutionary process. In sum, XCS performance can be improved by optimizing gradient-based approximation, classifier representation, and the evolutionary process.

Moreover, we introduced a new compaction mechanism to XCSF. The mechanism is based on *closest classifier matching* (CCM) plus condensation (neither mutation nor crossover in the GA application). In CCM, a fixed number of closest classifiers match, where closest is defined by the distance measure evolved for each classifier. CCM prevents the generation of holes in the function approximation surface during compaction. Meanwhile, condensation causes the propagation of well-shaped accurate classifiers and the deletion of overlapping inaccurate classifiers. The mechanism was able to decrease population sizes by often more that 80% hardly affecting performance accuracy. An additional greedy compaction algorithm, which iteratively deletes classifiers that overlap with low-error classifiers, was shown to be able to compact the population by often more than 90% on average—albeit with a slight accuracy decrease in non-differentiable or highly irregular functions.

Results showed that the improvements enabled XCSF to solve function approximation problems of up to seven dimensions with highly compact final representations. Moreover, XCSF was shown to be noise robust and able to generalize well to unseen problem instances. In general, it was highlighted that XCSF is a learning mechanism that clusters the problem space to ensure maximally accurate approximations in the experienced subspaces. It outperforms general clustering algorithms, such as Neural GAS (Martinetz, 1993), in function approximation tasks, and it approximates and partially outperforms more directed, iterative function approximation mechanisms published elsewhere (Potts, 2004, Schaal, Atkeson, 1998).

In conclusion, XCSF was shown to be a flexible, easily adaptable learning system, which is applicable to many types of predictive tasks, and particularly to tasks that can be approximated with local, partially-overlapping gradient-based estimates. Future work will evaluate the XCSF enhancements in datamining tasks as well as in reinforcement learning problems. Moreover, the mechanism is planned to be integrated into a cognitive systems architecture, in which the predictions manipulate neural gates in a recurrent neural network structure. In this case, the predictive capabilities are planned to be evaluated on a visual processing and attention focusing task, in which moving objects are tracked and their velocity as well as type of object will be extracted (See also the hierarchical visual tracking task described in deliverable 4.1). Albeit the accomplishment of this endeavour will most likely extend over the boundaries of the MindRACES project, the gained insights in the predictive capabilities and the high learning flexibility of XCSF will be invaluable for its success.

2.3 Policy Gradients (IDSIA)

Wierstra, Förster, & Schmidhuber (submitted) are working on a solution for deep memory POMDPs with recurrent policy gradients. The paper presents Recurrent Policy Gradients, a model-free reinforcement learning method that creates limited-memory stochastic policies for Partially Observable Markov Decision Problems (POMDPs) that require long-term memories of past observations. The approach involves approximating a policy gradient for a recurrent neural network by backpropagating return-weighted characteristic eligibilities through time. By the usage of a "Long Short-Term Memory" architecture, it was possible to outperform other reinforcement learning methods on two important benchmark tasks: double pole balancing task with incomplete state information and the long term dependency T-maze task. Furthermore, good results were shown in a complex car driving simulation task (TORCS racing car simulation).

The TORCS racing car simulation.

2.4 Reinforcement Learning for Robot Navigation (IDSIA)

Besides policy gradients, also reinforcement learning itself was further investigated. Zhumatiy, Gomez, Hutter, & Schmidhuber (2006) and also Bakker, Zhumatiy, Gruener, & Schmidhuber (2006) equipped a robot with a color camera and placed it into a room. The task was to find and move to a randomly placed unique colored cup in the room. The camera was mounted in front of the robot and looked a bit downwards. It had a very limited field of view in relation to the room. Therefore, the robot had to find the cup before it could move to the target position.

The controller of the robot translates sensor input data to robot movement commands. It is trained by different reinforcement learning methods. In Zhumatiy et al. (2006) the mean position of all camera pixels in a specific color range of the target object was used as input for the reinforcement learner. To reduce the huge amount of memory for the policy a "Piecewise Continuous Nearest-Sequence Memory (PC-NSM)" algorithm was used for general metrics over state-action trajectories. In Bakker et al. (2006) the visual information from the camera was preprocessed into a 5x4 binary grid, which represents the position of the cup in the camera image, if the cup is visible. To reduce the generally long training time for reinforcement learning algorithms for real robots, a probabilistic world-model was learned from less real robot experiments. This world-model was then used to make mental experiments on this model to train the controller with Prioritized Sweeping, which is an enhancement of the standard Q-Learning algorithm. The policy was applied with a high repetition rate during the learning process of the mental model and with a real time repetition rate in the physical world.

Camera image obtained by the robot: The target object, the yellow cup, is in sight. Detected target color pixels are indicated in blue. The small yellow cross marks the center of the cluster of target color pixels. This information is quantized using a regular 5 by 4 grid (white lines). Bold red lines indicate the grid cell corresponding to the quantized state information.

2.5 A Testbed for Neural-Network Models Capable of Integrating Information over Time (ISTC-CNR)

Stefano Zappacosta et al. (in press) propose a testbed for recurrent neural networks and related systems to integrate information over time. More in particular, the testbeds allow evaluating the capability of such models, and possibly other architectures and algorithms, of (a) categorizing different time series, (b) anticipating future signal levels on the basis of past ones, and (c) functioning robustly with respect to noise and other systematic random variations of the temporal and spatial properties of the input time series. The task is to scan an object or a wall while moving around it or along it, respectively. The recurrent network is trained to classify the object scanned, investigating prediction robustness, noise-robustness, and different aspects of generalization capabilities of the network in question. The paper also presents a number of analysis tools that can be used to understand the functioning and organization of the dynamical internal representations that recurrent neural networks develop to acquire the aforementioned capabilities. For example, to understand how they capture time regularities such as periodicity, repetitions, spikes, numbers, levels, and rates of change of input signals. Elman networks, leaky integrator networks, long-short term networks, and echo state networks are exemplarily introduced as suitable network candidates. The utility of the proposed testbeds is illustrated by testing and studying the capacity of Elman neural networks to predict and categorize different signals in two testbed instances: a wall task, in which two different wall patterns need to be distinguished, and an object task, in which three different objects are perceived. The testbed, possibly with additional action-information of movement type and speed in the future, seems to be a valuable tool to test and compare the capabilities of different time-series classification algorithms on somewhat real-world robotic classification tasks.

2.6 Artificial Immune Systems for Anticipation (AISA) (OFAI)

In the current status of the AISA implementation for the hunter and prey scenario there are two robots involved: On the one hand there is the Sony AIBO ERS-7 robot, which plays the role of the hunter and is controlled by an Artificial Immune Network architecture. On the other hand the MyBot robot, which is substituted by KURT3D in the real world scenario, which represents the prey, is controlled by a simple subsumption architecture (Brooks, 1991) that makes it move in a “foreseeable” and ordinary motion (e.g. a circular trajectory or wall following).

In the scenario the AIBO robot has to catch its prey MyBot, which provides the reward upon success. In accomplishing this task the hunter has to anticipate the prey’s movement trajectory and try to get into its way in order to catch it. The complex behaviors learned and developed in this task perfectly show the anticipatory capabilities of the AISA framework and their suitability to robot control. The complexity of the scenario can be varied in two ways:

1. By changing the environment. E.g. by creating hiding places for the pray, where the hunter cannot see it; by introducing rooms; or, more generally, by making the topology more complex so that clear view for the hunter is omitted.
2. By increasing the degree of complexity of the movement strategies of the prey. If the prey follows a predictable and constant trajectory it is of course much easier to catch, as if it’s tumbling or alike.

In the beginning, the robotic agent observes the prey and consequently tries to intercept it. It does this by anticipating the movements of the prey and since it cannot move much faster than the prey, by finding a shorter way to an anticipated location where it can bar its way.

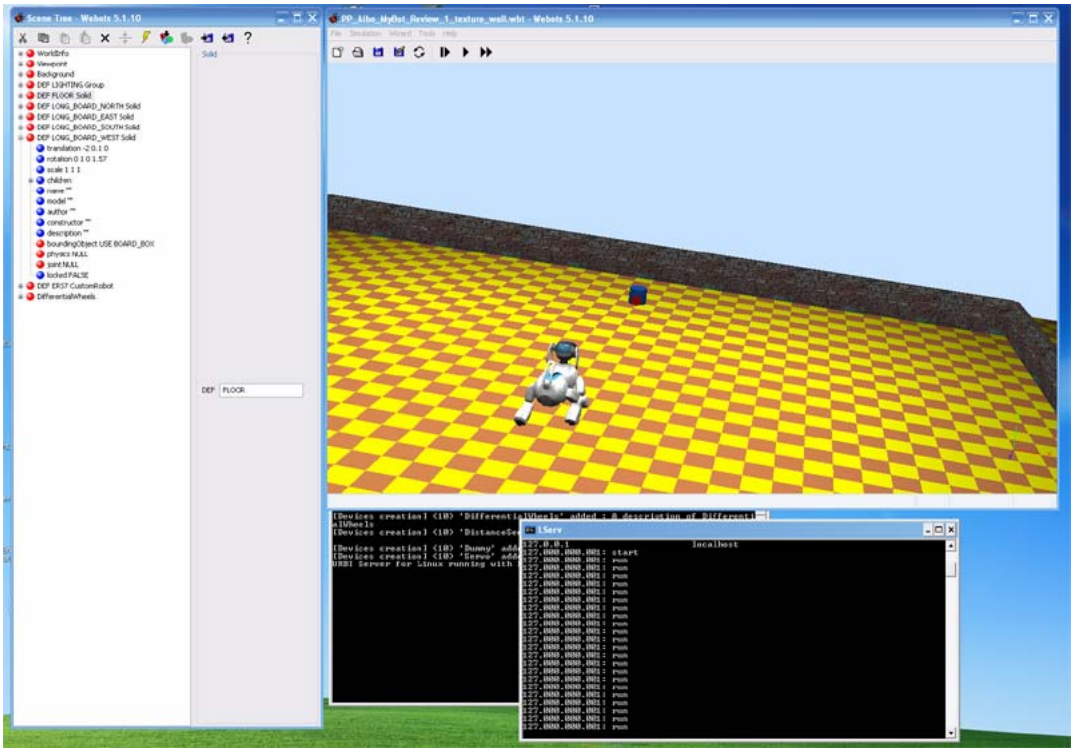
In the future the scenario will be extended, wherefore the robotic agent might then observe and pursuit the prey and learn typical “hiding places”, so that later on when hunting and trying to catch a prey it may move to anticipated “hiding places” and find prey much easier than by trial and error search.

2.6.1 Programming and Simulation Environment

To ensure a feasible amount of platform independency we use Java as the programming language of our architectural framework and Eclipse as software development kit as a wrap around it. All parts of

the robot control architecture are layered and AISA only accesses abstract sensors and actuators. This enables us to easily exchange any robotic platform into another by just changing codes for robot control and sensor access only at the lowest layer and leaving the AISA architecture unaffected and operational at all times.

As the AISA architecture should work well both with real robots and in simulation, we needed a programming and simulation environment providing us with an open robot interface.

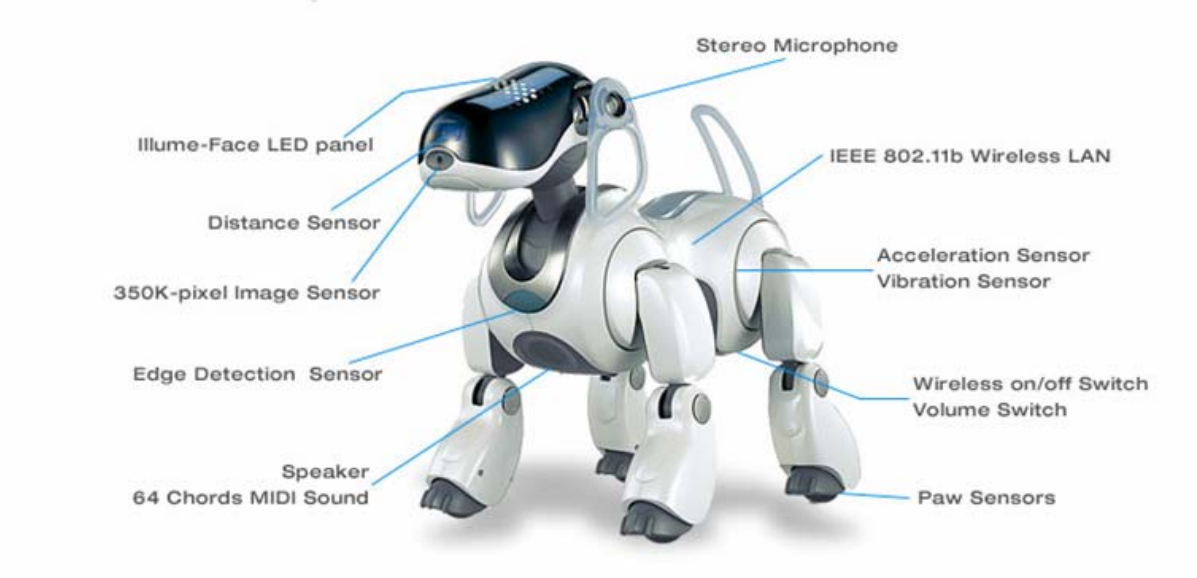


The Webots 5 programming environment.

We found Cyberbotic's Webots 5 Simulator (Michel, 2004) to be suitable for our needs: on the one hand it includes a very good implementation of the AIBO ERS-7 robot that we use as our main robotic platform for all our experimentation and on the other hand Webots supports the Universal Real-Time Behaviour Interface (Baillie, 2005), which enables us to comfortably access any real robot we need to include in our test bed (e.g. MyBot). Additionally Webots allows us to model the simulated environment in a slightly simplified version of the Virtual Reality Modelling Language (VRML, Carey *et al.*, 1997), which enables us to create environments of nearly unlimited complexity in a very convenient and rapid prototyping way.

2.6.2 Robot Platforms

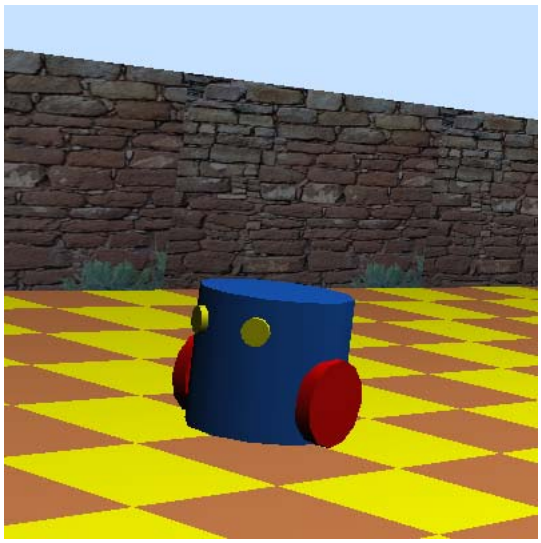
We perform our experiments in simulation (for rapid prototyping) as well as on real robots and use three different robotic platforms in the different environments. Regarding the hunter prey scenario it is also planned to switch the roles of hunter and prey between these robots in order to evaluate the platform independency of the AISA architecture, the Sony AIBO ERS-7. Currently OFAI is primarily using the Sony AIBO robotic dog platform (**A**rtificial **I**ntelligence Ro**bo**t) and we mainly use AIBO in our research and as the hunting agent in all experiments regarding our current state of development.



Sony AIBO ERS-7 robotic platform.

Although AIBO was created initially as a robot for home entertainment, it is being used by a lot of researchers interested in low-cost programmable robot platforms. Unfortunately the production of AIBO robots was discontinued in spring 2006. Still service and support was prolonged for the next five years and most people continue using AIBO as they did before.

In the simulated environment, we use the MyBot robot as the prey. MyBot is a very simple two wheeled robot with two distance sensors. It is included as an example virtual robot in the Webots 5 programming environment. It is programmed for obstacle avoidance and, as long as it is not disturbed by sensory perceptions, it mainly follows a circular movement trajectory.



The MyBot virtual robotic platform.

The robot used as prey in our real world experiments is a 6-wheeled KURT3D robot (Surmann, H., & Pervoelz, K., 2003) originally designed for sewage pipe inspection. Its dimensions are 45 cm in length, 33 cm in width, and 51 cm in height (including laser range finder) and it has an approximate net weight of 10.4 kg. The robot carries a laptop for control and a 3D laser range finder that increases the weight to totally 22.6 kg. KURT3D is additionally equipped with two Logitech VGA-Cameras and may operate for about 4 hours with one battery charge. An embedded 16-Bit CMOS microcontroller is used to control the motor and lower sensors. The maximum controlled velocity the robot may reach using its two motors is 14.4 km/h.

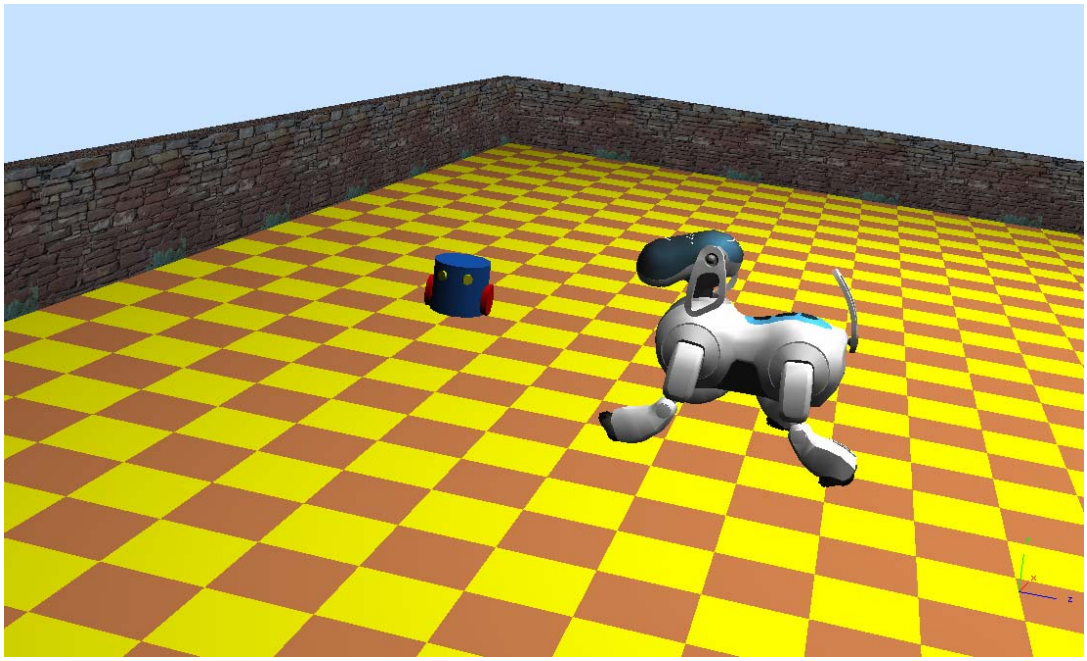
In our experiments from the great number of sensors it is equipped with we only use its 12 infrared and 2 ultrasonic sensors. KURT3D is also programmed to go around in circles and avoid obstacles, if it encounters them. Of course the potentially high velocity the robot may reach makes it an interesting prey for our AIBO robot.

KURT 3D and AIBO in the test bed

2.6.3 Test bed

All experimentation is done in simulation using version 5 of Cyberbotics WeBots mobile robot simulation and in a real world environment.

As test bed for both environments in our scenario we use a wide plain area of wooden floor which is surrounded by wooden boards with a height of 15 cm, and which are arranged in different manners (e.g. rectangular, circular). Typical test beds are depicted in following figures for the simulated environment and the real world environment.



AIBO and MyBot in the simulated environment.

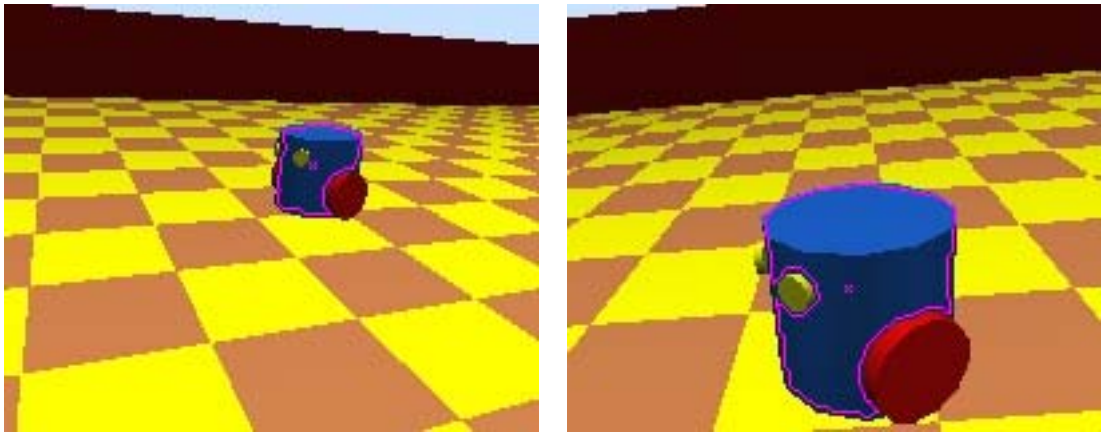
AIBO and KURT3D in the real word environment.

2.6.4 Sensor Input, Sensor Pool & Abstract Filters

From the large number of sensors the AIBO is equipped with we only use a smaller subset, namely the two distance sensors (nose and chest), from which the chest sensor is only a binary sensor for edge detection, and the camera image has a resolution of 208x160 pixels. Since the raw camera image in terms of complexity would be far too much information to the artificial immune network and respectively for the encoding into an *Epitope*, we define several virtual sensors (i.e. filters) representing the current visual perception.

To make the sensor input suitable for processing with AISA, i.e. to be able to construct a genetic string, which encodes an *Epitope, Paratope* or *Idiotope*, it is feasible to have integer values derived from all sensory inputs including the streaming image data.

In addition to the two values of the distance (infrared) sensors, which are converted to a binary value range from 0, meaning low, to 1, meaning high, we use a set of abstract filters that encode the current visual input into the requested discrete integer values. All the image filters are generated by a blob detection and evaluation algorithm, which translates the image into numeric values namely the mass value of the detected blob, corresponding to the size of the object and the centre of gravity of the object, which corresponds to its position.



Application of the blob filter to an AIBO camera image from the simulated environment.

In our current implementation we use three different virtual image sensors. For all this filters a value of 0 means that there is no blob within the visual range:

1. *getHorizontalBlobPosition*, which returns the horizontal position of the centre of gravity of a detected blob and is mainly used to detect motion in the horizontal perception axis. Further its history values may be used to derive some kind of speed of the perceived blob.
2. *getVerticalBlobPosition*, which returns the vertical position of the centre of gravity of a detected blob and is mainly used to detect the motion in the vertical perception axis. It may additionally be used to derive the distance of the blob, and of course its speed as well.
3. *getBlobSector*, which returns the position of the centre of gravity of a detected blob in terms of an image sector value. I.e. the image is split up into nine sectors (numbered from left to right and from top to bottom), and the centre of gravity of the detected blob is projected into those sectors. This filter in a way accumulates the first two sensors, while being less accurate than these.

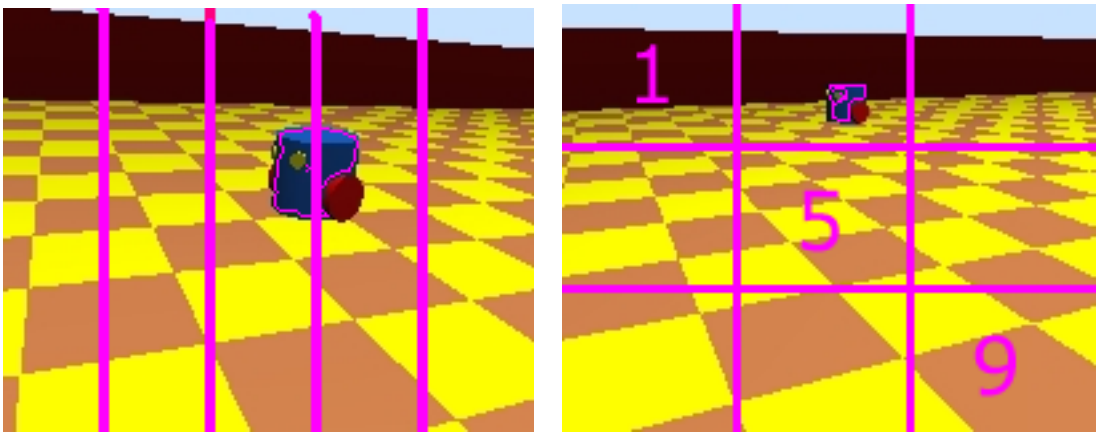
Table 1 summarizes the sensors and filters we use in our current anticipatory control architecture. As described in more detailed in the next section, all genetic strings are composed out of sensor and filter values.

Type	Name	Description	Value Range
Distance Sensor	*Infrared1*	Nose Distance Sensor	0 … 1 (Low, High)

Distance Sensor	*Infrared2*	Chest Edge Detection Sensor	0 … 1 (Low, High)
Image Filter	*getHorizontalBlobPosition*	Returns the horizontal position of the centre of gravity of a detected blob	0 … 5
Image Filter	*getVerticalBlobPosition*	Returns the vertical position of the centre of gravity of a detected blob	0 … 5
Image Filter	*getBlobSector*	Returns the position of the centre of gravity of a detected blob as an image sector	0 … 9

Summary of Sony AIBO ERS-7 sensors and their according filters used in AISA.

Although we find the remaining sensors (paw sensors, stereo microphone, acceleration sensors) not being necessary for solving the tasks in the current hunter prey scenario, which is somewhat vision dependent, it seems to be evident that in more extended and complicated scenarios the usage of stereo audio could be valuable and necessary to effectively find a hiding prey. It is therefore planned to include the audio sensors (using suitable audio filters) in future, refined scenarios.



Application of the *getHorizontalBlobPosition* (left) and the *getBlobSector* (right) visual filters to a AIBO camera image from the simulated environment.

2.6.5 An Artificial Immune System for Anticipation

Using the robotic platforms, test beds, and scenarios described before we developed an anticipatory control architecture, which is largely based on the artificial immune system metaphor, originally suggested by Farmer (Farmer, 1986). Farmer proposed a possible relationship between biological immunity and computing by comparing natural immune systems with the principles or adaptation and machine learning. Since then a vibrant and steadily growing community, which translated the metaphor to a large number of application domains, has grown in this research area (Hart & Timmis 2005).

The artificial immune system approach we use in our anticipatory control architecture was particularly inspired by Jerne's immune network theory (Jerne, 1974), which proposes that the memory and learning capability of the immune system is not only founded by the interaction of B-lymphocytes (a type of antibody) with antigens, but also by B-lymphocytes interacting with each others, even in the absence of foreign antigens. While this view is subject to debates among immunologists, it proved to be successful for several applications domains including ours.

The antibody represents the immune system‘s reaction to specific antigens (sensor input). The antibody’s antigen-identifier – the *Paratope* – fires and executes a specific action, if its affinity to the antigen (and therefore *Epitope*) is above a certain threshold (in biological immune systems it would bind to and mark the antigen). If more than one antibody would be able to deal with the current antigen, from the ones which share the best affinity a winner is randomly chosen.

In our case we defined a specific set of simple and atomic actions, which are to some extent independent of the robot platform. Over time the immune system learns how to react to specific sensor input. The atomic platform independent actions we use are:

1. *moveForward*: the robot move forward for a specific time interval
2. *moveBackward*: the robot moves backward for a specific time interval
3. *turnLeft*: the robot turns left for a specific time interval
4. *turnRight*: the robot turns right for a specific time interval

Additionally we defined some atomic actions that require the robot to have an actuator:

1. *Kick*: the robot kicks an object in front of it using its actuator
2. *Poke*: the robot pokes an object in front of it using its actuator

All these actions are defined on the platform abstraction layer of URBI (see also section 4.1 Programming and Simulation Environment) and only executed by the control architecture. This enables us to use the same architecture on different robots by only having agent dependent libraries that link the actions to the corresponding low level commands of the hardware architecture. The atomic action *moveForward*, if for example executed on the AIBO would involve the control of all the hinges of the four legs in an adequate manner, while on a wheeled robot it would be just necessary to activate the motor controller of the two axes. Still on the control level only the action command to let the robotic agent move forward has to be executed.

2.6.6 How Anticipation Comes Into Play

According to Jerne’s immune network theory (Jerne, 1974) we are missing one ingredient that enables our antibodies to interact with each other. This part of the antibody is called the *Idiotope* and brings also expectation and anticipation into play.

Consequently each antibody has an expectation (the *Idiotope*) about the results of the action it performed and therefore determining those antibodies which will be able to fire next. This linkage to one or more successor antibodies is realised via the *Paratope-Idiotope* affinity of all antibodies to all others. Antibodies that are linked may stimulate each other and suppress other antibodies. E.g. an antibody that never gets activated and therefore does not feasibly correspond to sensor input, will receive permanent suppression and after some time be dropped out of the network. On the other hand antibodies that often get activated will receive promotion and distribute this promotion also to its successors. As a result the links between these antibodies get stronger and paths corresponding to higher level actions emerge.

As the robot acts in a dynamic environment with the higher goal to catch a prey, the underlying artificial immune system control architecture and the network of antibodies is cycling through a continuous process of interaction and reinforcement that enables the network to evolve, develop, and adapt over time. The successive learning iterations that are performed until some stopping criteria are met will be shortly described in the following:

1. Obtain and process the current data conditions (antigen)
2. Antibody that fits best is chosen (Stimulation) using a winner takes all regime
3. The winning antibody's action is performed
4. Evaluate the antibody and its performed action (Reinforcement)
5. Compute and update all network links
6. Clone winning antibody (Hypermutation)
7. Remove dead links and dead antibodies

A learning iteration starts with the processing of sensory input. First, the current antigen, that is, the input string, is generated from the sensor values and filters. Next the antigen-antibody interaction commences by the determination of affinity for each antibody and the selection of the antibody with highest affinity by a winner takes all regimes. If there are antibodies having equivalent affinities one of them is randomly chosen. After the antibody was successfully chosen its action is performed.

The determination of the closest antibody then triggers an antibody-antibody interaction, which essentially co-activates structurally similar antibodies as described above. Accordingly new antibody concentrations are calculated for the whole network. These are rising for antibodies that have a high affinity to the winning one and decreasing if they have low or negative affinity to it. Next, the antibody chosen in the last time step is evaluated. Hereby, reinforcement learning techniques are applied respecting current internal drives. Essentially, the concentration is adapted again by a cumulative value with respect to conformance of motivations such as greed, hunger, or explore, and also the quality of predictions.

After antibody evaluation, the evolutionary component is triggered. Here antibodies are selected for reproduction respecting their current affinity measure. The selected antibodies are cloned and hyper-mutation is applied. Similarly *Apoptosis* removing old and useless (concentration below a certain threshold in relation to total sum of antibodies) antibodies from the network is applied. Additionally memory cells are contained. Antibodies become memory cells once their concentration exceeds a certain concentration threshold. Such high concentrations essentially indicate that the antibody is highly useful and consequently should be protected from deletion. Memory cells cannot be deleted and will stay in the population until program termination.

This concludes the loop and the algorithm starts over again in the next time step.

2.6.7 Experimentation, Preliminary Results and Future Work

Using the previously described architecture we performed several experiments in the simulated environment and are constantly involved into porting and evaluating the architecture in the real world scenario. In the following we report about the promising results of our recent experiments and give an outlook to potential optimisations and future work where applicable.

A very interesting variable of the artificial immune network is of course the antibody population size, which may vary over time and which can be constrained by defining a maximum number of antibodies that the network is allowed to consist of. If the population size is too small or too large, the network may never stabilise, additionally if the network is too small, learned behaviours will permanently get deleted from the network and have to be learned again.

In our experiments we used varying initial population sizes starting from 10 to 100 antibodies and were limiting the maximum size from 100 to 1000 antibodies. A maximum number of 100 antibodies turned out to be feasible and sufficient for the task of catching MyBot in our current scenario. About one third of these antibodies gets heavily used then and is turned into memory cells over time. In the simulation the network gets stable after about 10 minutes of learning. Thereafter, AIBO is able to catch MyBot in a stable and comprehensible manner.

Since the visual data in the real world scenario does not produce as stable and predictable results as the simulation does, the network size for the real robot and its control architecture has to be larger

and the process of stabilisation lasts longer. One important part of our work in the near future will be further experimentation on this issue.

Since the network's reactions to and interactions with the environment are only canalised through the *Epitopes*, the definition of the genome is crucial for any artificial immune system and network to produce reasonable behaviour. We evaluated several antigen genomes of different lengths and also paid much attention to the construction of the visual filters and their application. In our current hunter prey scenario, it turns out that the *getHorizontalBlobPosition* filter including its history is most important and the strategy the robot develops heavily depends on its values.

AIBO seems to learn a strategy like:

a. "If I can see the prey, then I turn into the direction it is moving until I cannot see it anymore."
b. "If I cannot see it anymore, I move forward."

This strategy leads AIBO to anticipate MyBot getting into view again and being closer at this time, and consequently getting in front of the prey and catching it.

Real world experimentation shows that visual noise that makes its way through the filters leads to a reduced reliability of the single filter values. The resulting antibodies need to depend on more than one filter value at a time. In this course, the *getBlobSector* gets more importance and seems to be used in addition to "verify" the presumption about the situation the agent is currently exposed to (*Epitope*).

2.7 Goal-Initiated Behavior System Comparison (ISTC-CNR & UW)

ISTC-CNR and UW recently published a joint investigation on the differences between the Ideomotor principle from the field of psychology and the test-operate-test-exit (TOTE) system from cybernetics. The following text provides background and overview of this work (Pezzulo, G.; Baldassarre, G.; Butz, M.V.; Castelfranchi, C. & Hoffmann, in press).

Intelligence of complex organisms, such as humans and other apes, resides in the capacity to solve problems by working on internal representations of them, that is, by acting upon ``images" or ``mental models" of the world on the basis of simulated actions (``reasoning"). These capabilities require that internal representations of world states, goals and actions are intimately related. With this respect, accumulating evidence in psychology and neuroscience is indicating that anticipatory representations related to actions' outcomes and goals play a crucial role in visual and motor control (Hesslow, 2002). As suggested by the discovery of mirror neurons (Rizzolatti et al., 1996), representations are often action-related and are thus grounded on the representations sub-serving the motor system. Barsalou (1999) and Grush (2004) try to provide unitary accounts of these phenomena respectively proposing *perceptual symbol systems* and *emulation* theories of cognition. In a similar vein, Hesslow (2002) proposes a *simulation hypothesis* according to which cognitive agents are able to engage in simulated interactions with the environment in order to prepare to interact with it. According to Gallese (2000): ``To observe objects is therefore equivalent to automatically evoking the most suitable motor program required to interact with them. Looking at objects means to unconsciously `simulate' a potential action. In other words, the object-representation is transiently integrated with the action-simulation (the ongoing simulation of the potential action)".

Recently anticipatory functionalities have been started to be explored from a conceptual point of view (Butz et al., 2003, Castelfranchi, 2005, Roy, 2005) as well as from a computational point of view (Drescher, 1991, Butz & Hoffmann, 2002, Butz, 2002, Pezzulo & Calvi, 2006, Wolpert & Kawato, 1998). This paper contributes to this effort by analyzing two important now ``classic" frameworks of goal-oriented behavior, namely the *ideomotor principle* (IMP), and the test operate test exit model (*TOTE*).

The IMP and the TOTE can be dated back in their origin for decades if not centuries. The IMP, which was proposed multiple times during the 19th century within the psychological literature (Herbart, 1825, James, 1890), hypothesizes a bidirectional action-effect linkage in which the desired (perceptual) effect triggers the execution of the action that previously caused it. The TOTE, introduced within the field of cybernetics (Miller, Galanter, & Pribram, 1960), proposes that goal-oriented action control is based on an internal representation of the desired world's state(s) with which the current world's state is repeatedly compared to in order to direct actions.

The first goal of the paper is to provide a comprehensive introduction to both the IMP and the TOTE and to highlight their similarities, differences, and drawbacks in explaining anticipatory goal-oriented behavior. The second goal of the paper is to analyze, at an abstract level, three computational architectures, which implement various different features of the IMP and the TOTE in distinct ways. The architectures are only reviewed here, while the reader is referred to specific papers for details. The analysis aims at exemplifying and clarifying the principles underlying the IMP and the TOTE. It is intended to serve as a starting point for future research on the investigation of anticipatory goal-oriented behavioral mechanisms. A final discussion concludes the paper with an outlook of the most important challenges that the two principles pose to cognitive science.

2.8 Goal-initiated behavior: Experimental results for the Robotic Eye-Arm System (ISTC-CNR)

This section is about the performance of the Robotic Eye-Arm architecture tested on a reinforcement learning task with various flavours of the system (with or without the real arm, with or without some pre-learning) and with different input configurations. The value of the use of various bio-inspired architectural choices, like developmental learning and population encoding, is verified from the results. Other interesting findings are about the influence on the performances of nonlinear coding of the visual input and the influence on the performances of the real robotic arm versus the simulated arm.

The remaining sub-sections illustrate:

- The task that will be used to test the Robotic Eye-Arm System
- The model architecture
- Description of the eye-arm robotic system and of the environment
- The descriptions and the results of the following experiments:
    - Experiment 1: Test Of Speed of Learning Phase 4 (Reinforcement Learning)
    - Experiment 2: Accuracy of Reaching After Learning Phase 4(Reinforcement Learning)
    - Experiment 3: Performance without learning phase 3 (pre-training)
    - Experiment 4: Learning single patterns (Reinforcement Learning)
    - Experiment 5: Performance with the real arm (Reinforcement Learning)

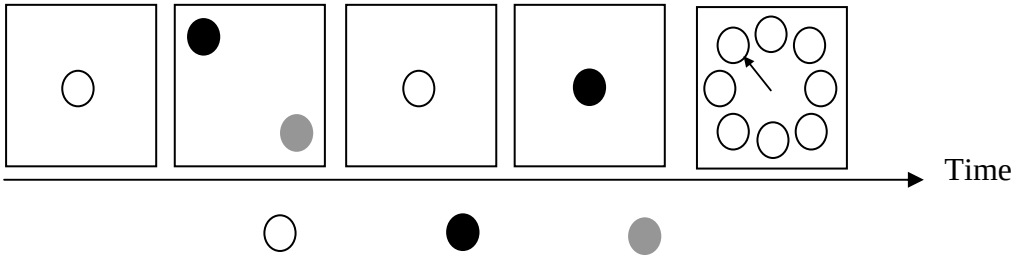
2.8.1 The task

The robotic architecture presented here will be tested using a simplified version of the “discrimination and reaching task” used by Cisek & Kalaska (2005) to carry out physiological recordings in monkeys’ premotor cortex. The task is composed of five phases:
1) *center-hold time*: the monkey’s hand is positioned on a manipulandum at a central starting position of an horizontal plane, and a green cue circle appears at the center of a screen set in front of the subject;
2) *spatial cue*: a red and a blue circle (with a 5 cm radius) appear on the screen at two opposite positions of eight possible target locations distributed around a circle;
3) *memory*: a green cue circle appears again at the center of the screen;
4) *color cue*: a color cue, either red or blue, appears at the center of the screen: this non-spatial cue signals which of the two memorized color-coded spatial cue locations is the target that the monkey should reach;
5) *go signal*: eight green circles appear at all the possible target locations: if the monkey reaches the target position that matches both one of the two spatial cues *and* the color cue, it receives a reward.

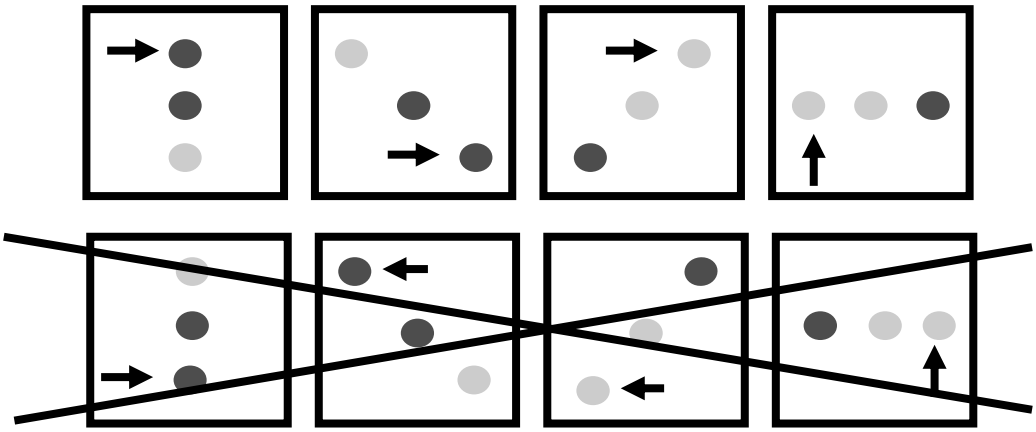
In the simplified version used for the test of the robotic arm-eye system there is only one image shown to the system per trial. It is composed of the spatial cue and the color cue.
The task has a number of advantages with respect to testing the Robotic Eye-Arm System proposed here:

- The experiment has already been tackled with success by ISTC-CNR with a bio-inspired neural architecture (Ognibene et al., 2006).

- The task has been taken from a neuroscience paper (Cisek & Kalaska, 2005) that describes interesting data, physiological and behavioral, related to monkeys engaged with the task: these data can be compared with the results obtained with the computational architecture proposed here.
- The simplified task is not as computationally complex as the original one (Cisek & Kalaska, 2005, Ognibene et al., 2006) so the set of "x-or" sub-problems and the integration of information in time that required a lot of training trials are removed, so to be usable on a real robot.
- The task allows simplifying the hard (and out of scope) problem of object's recognition as targets and cues have distinct colors.



The five phase of the task represented by the screen images that the system perceives. The arrow in the last fifth image indicates the movement the system has to perform in order to get the final reward (see text for details).



Simplified version of Cisek & Kalaska task: synchronous image (no memory needed); only 4 patterns out of 8 (No x-or problem, perceptron is enough).

2.8.2 The model architecture

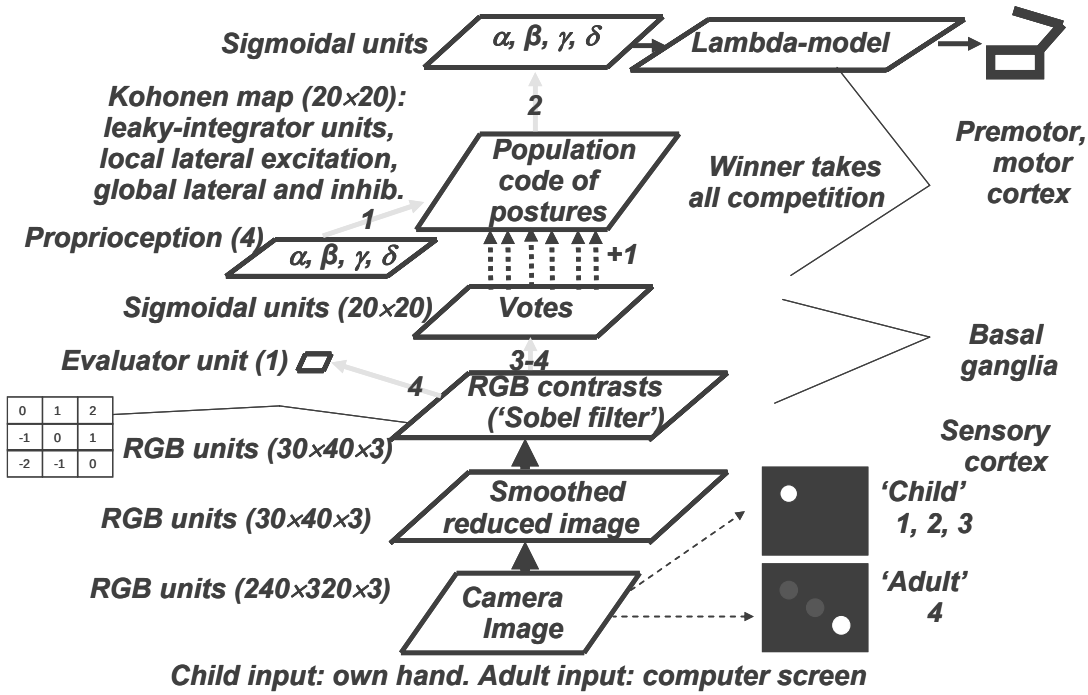
The models of ISTC-CNR focus on control and tackle the following anticipatory functionalities:

- Internal representations of goals as anticipated states that govern the behaviour of the system (the activation of these representations lead the system to act in the world in order to increase the probability that they will be accomplished: cf. Pezzulo et al., 2006).
- Selection of future goals on the basis of a winner-take-all dynamic competition between goals. The integration of information in time is anticipatory with respect to future overt behaviour.
- Reinforcement learning capability based on the anticipation of future discounted rewards.

The architecture of the model is shown below. The figure indicates the components of the architecture and the corresponding brain parts (see Kandel et al., 2000). The functioning and learning processes of the components of the architecture are now explained in further detail.

The *retina*'s units are activated by the images on a screen thorough a webcam. The retina is composed of 40x30 units for the red, blue and green components. The image from the weights is filtered thorough a smooth rescale filter and Sobel filter to detect edges.

The *actor-critic* components controlling the arm are a neural implementation of the actor-critic model (Sutton and Barto, 1998). The *actor* (*basal ganglia's matrix*) is a two-layer feed-forward neural network with 20×20 input units, that correspond to the units of the retina, and 20×20 output units. The output units have a Sigmoid transfer function with activation y_j and each has a



The neural components of the architecture with the corresponding brain areas on the right side.

topological one-to-one connection (with weights equal to $\upsilon = +1$) with the posture controller's input units. The *critic* (*basal ganglia's striosomes* and *substantia nigra pars-compacta*) is mainly composed of a neural network ("evaluator") having a linear output unit. At each step t this output unit produces evaluations V_t of perceived states, and the critic uses couples of successive evaluations, together with the reward signal R_t, to compute the *surprise* signal S_t (*dopamine*):

$$S_t = \left(R_t + \gamma V_t\right) - V_{t-1}$$

where γ is a discount factor ($\gamma = 0.3$). The surprise signal is used for training both the actor and the evaluator (see below).

The *accumulator units* (*premotor cortex*) form a 2D 20×20 map, have all-to-all lateral inhibitions, and have local excitations that decrease with distance on the map. The units engage in a many-winner competition on the basis of the signals ("votes") that they receive from the actor's output units via the one-to-one connections (Usher and McClelland, 2001; Schall, 2001). In particular, they behave as *leaky-integrators* and have an activation a_j as follows:

$$a_{j\,t} = \max\left[\left(a_{j\,t-1} + \frac{dt}{\tau}(D)\right), 0\right]$$

$$D = \chi\left(-\delta a_{jt-1} - \iota \sum_{l,l\neq j} a_{lt-1} + \eta \sum_{l,l\neq j} e_{jl} a_{l\,t-1} + \upsilon y_j + \varepsilon_{jt} + \varepsilon_{jc}\right)$$

where τ is a time constant, corresponding to 1/10s, dt is the integration time step ($dt = 0.05$ 1/10s, so $dt/\tau = 0.05$; a_j is numerically updated every 0.005 s), χ regulates the speed of the dynamics ($\chi = 1$), δ is a decay coefficient ($\delta = 0.1$), ι regulates the all-to-all lateral inhibition ($\iota = 0.15$), η regulates the

local lateral excitation (η = 1), e_k represents the fixed weights of the lateral excitatory connections (e_k is set to 0.4 for neighboring units along the x/y-axes directions, to 0.2 for neighboring units along the diagonals, and to zero for all other units), ε_{jt} is a noise component that ranges over [-0.1, +0.1] and varies in each cycle, ε_{jc} is a noise component that ranges over [-0.25, +0.25] and is constant for time intervals c randomly drawn from [0, 5] s (ε_{jc} is important for exploration of reinforcement learning as various ε_{jt} tends to sum to zero over many steps). When the activation a_j of one accumulator unit reaches a threshold T (T = 1.9), the total activation of accumulator units is *normalized to 1*, their dynamics is "frozen", and the execution of a reaching sensorimotor primitive is triggered.
The *posture controller* has an input-unit layer corresponding to the accumulator units and two Sigmoid output units, with activation d'_k, that range over [0, 1] (*motor cortex/spinal cord neurons*). The activations of these output units are remapped onto the arms' angles and form the commands issued to the posture servomechanism in terms of arms' desired angles (posture). It is important to notice that these desired angles are generated by the *cluster* of accumulator units that are active at the end of the many-winner competition. This implies that the target of the executed sensorimotor primitive is a *mixture* of the targets "suggested" by all active units: this *population encoding* (Pouget et al., 2003) allows the arm to cover the whole continuous space of postures.
The *posture servomechanism* is a hardwired closed-loop controller (*Golgi tendon-organs*, *muscle-fiber afferents*, and *spinal cord*, cf. Shadmehr and Wise, 2005) that issues commands to the arm's actuators (*muscles*) on the basis of the desired-posture command received from the posture controller.

Learning phases related to the arm. The learning processes take place in two phases, the *childhood phase* (three processes) and the *adulthood phase* (one process). Now we first present an overview of these learning processes and then describe them in detail.
During the childhood phase the system performs motor babbling: in practice the arm randomly varies its joints' angles, with changes $\Delta d'_k$ belonging to [-10, +10] degrees, without violating the joints' constraints. Motor babbling is used for performing three learning processes. The first two processes allow the system to learn to perform sensorimotor primitives, in particular: (a) to train the 2D map of accumulator units, through a Kohonen algorithm (Kohonen, 2001), to represent the postures perceived by the proprioceptive units d_k (during the childhood phase the proprioceptive units, the accumulator units, and their connections, function as a Kohonen network); (b) to train the posture controller, through a Widrow-Hoff algorithm (Widrow and Hoff, 1960) (the generalized "delta rule"), to return as output the arm's angles corresponding to postures encoded in the Kohonen map. These two training processes lead the whole network formed by the Kohonen network and the posture controller to implement an "auto associative" function (i.e., the arm's angles encoded in the proprioceptive units are returned by the postural controller's output units). This whole network allows the system to recode postures, at the level of accumulator units, in an expanded format suitable to perform actor-critic reinforcement learning (Sutton and Barto, 1998). Notice that suitable *population encodings* at the level of the accumulator units allow the system to select *any posture* in the *continuous space* of postures: this is precisely what the actor-critic components learn to do while solving reinforcement-learning reaching tasks in the adulthood phase.
With the third learning process of the childhood phase the system's actor learns, through a Widrow-Hoff algorithm, to associate the point in space where the retina sees the arm's "hand" (i.e., the forearm segment's tip) with the activation pattern of the Kohonen map's units corresponding to such point (pattern caused by the arm's perceived angles). With this training, the actor acquires a bias to select sensorimotor primitives that drive the arm's hand to points in space corresponding to the retina's active units. This bias makes reinforcement learning performed during the adulthood phase quite fast notwithstanding the fact that the continuous space of postures is quite large. Note that two simplifying assumptions allow obtaining this result: (a) the retina does not perceive the arm and hand in the adulthood phase; (b) retina's units activated by the hand in the childhood phase are activated by the screen patterns in the adulthood phase.
During the adulthood phase the system learns by trial-and-error to accomplish Hikosaka's task. The actor-critic model used to this purpose has been suitably modified to be capable of selecting "actions" represented with population encodings. The four learning processes are now illustrated in detail.

Childhood phase: training of the Kohonen network. During the childhood phase, while the system performs motor babbling, the accumulator units receive input signals from two input units, having

activation d_k, that encode the arm's current angles (remapped in [-1, +1]: this information is thought to be returned by proprioceptive sensors located in the muscles, e.g. *Golgi tendon-organs* and *muscle-fiber afferents*, Shadmehr and Wise, 2005). An extra pseudo input unit is used to perform a "z-normalisation" of the input pattern: this is a normalization that preserves size information (Kohnen, 2001). The accumulator units are trained with a Kohonen algorithm (Kohnen, 2001) that allows them to develop representations of the arm's angles in their weights. The output units give place to a winner-take-all competition: the unit with the highest activation potential activates with 1 ("winning unit"), while the other units activate at levels decreasing with their distance from the winning unit on the basis of a Gaussian function. In particular, the activation a'_j of the unit j and the rule to update its weights w_{jk} are as follows:

$$a'_j = \exp\left[-\frac{h_{fj}^2}{\sigma^2}\right] \qquad w_{jk\,t} = w_{jk\,t-1} + \phi\, a'_j \left(d_k - w_{jk\,t-1}\right)$$

where h_{fj} is the distance on the map between the unit j and the winning unit f (h_{fj} = 1 for two contiguous units), σ is the standard deviation of the Gaussian function (σ = 1), ϕ is a learning coefficient (ϕ = 0.01). Note that the Kohonen algorithm uses a *winner-take-all* competition to activate the accumulator units instead of the *dynamic competition* reported in equation 3, used in the adulthood phase: indeed, the former tends to lead to an activation of the accumulator units that approximates the steady state activation that the same units would get through the latter (Kohnen, 2001).

Childhood phase: training of the posture controller. The posture controller is trained on the basis of a direct inverse modeling procedure (Kuperstein, 1988) that exploits the random movements $\Delta d'_k$ produced by motor babbling as follows: (a) the arm's angles are perceived and categorized by the Kohonen net; (b) a Widrow-Hoff algorithm (Widrow and Hoff, 1960, learning rate = 0.3) is used for training the posture controller's weights w_{kj} to associate the Kohonen-map units' activation (input pattern) with the angles d'_k caused by the random movements considered as desired output.

Childhood phase: pre-training of the actor. Through this pre-training, based on a Widrow-Hoff algorithm, the actor's weights w_{ji} are trained to associate the position of the hand perceived with the retina (input pattern $\boldsymbol{x}$) with the corresponding posture (desired output $\boldsymbol{a}'$) encoded in the Kohonen map (learning rate 0.1).

Adulthood phase: actor-critic's reinforcement learning. During the adulthood phase, the actor-critic component is trained to solve the task by reinforcement learning. During training, R_t is set to 1 when the arm reaches the two targets of any set of the hyperset in the correct order, and to 0 otherwise. The *evaluator* is trained after the selection and execution of a whole sensorimotor primitive (the primitive terminates when the arm reaches the desired posture selected by the posture controller). In particular its weights w_i are trained, on the basis of a Widrow-Hoff algorithm (learning rate ψ = 0.6) and a *TD-rule* (Sutton and Barto, 1998), as follows:

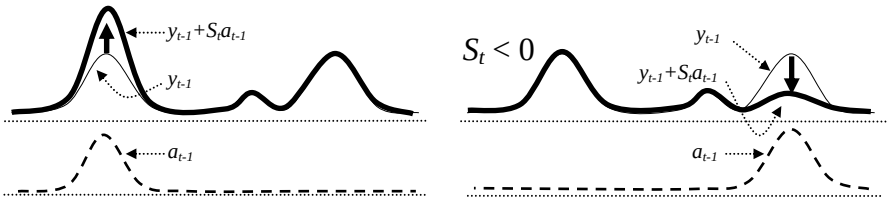
$$w_{it} = w_{it-1} + \psi S_t x_{it-1} = w_{it-1} + \psi\left(\left(R_t + \gamma V_t\right) - V_{t-1}\right)x_{it-1}$$

Through this learning process, the evaluator's evaluations V_t of the perceived states $\boldsymbol{x}_t$ tend to become higher for states corresponding to postures "closer" to reinforced states, and to form a gradient over the space of postures. The *actor* uses this gradient to learn to select highly rewarding sequences of primitives (Sutton and Barto, 1998). In particular the actor updates its weights w_{ji} with a Widrow-Hoff algorithm (learning rate ζ = 0.6):

$$w_{jit} = w_{jit-1} + \zeta\left(\left(y_{jt-1} + S_t a_{jt-1}\right) - y_{jt-1}\right)\left(y_{jt-1}\left(1 - y_{jt-1}\right)\right)x_{it-1}$$

where $(y_{jt-1}(1-y_{jt-1}))$ is the derivative of the Sigmoid function. The functioning of this learning rule is illustrated below. The rule tends to update only the weights of the units of the "winning cluster"

because the activation a_j of other units tends to be zero at the end of the race. The votes of the winning units are decreased or increased in correspondence of respectively positive and negative surprises.



Effects of the actor's learning rule of equation 6 illustrated with a scheme relative to a 1D layer of actor's output units (horizontal axis). Left: with a surprise St > 0, the actor's votes yt-1 (upper graph), that caused certain accumulator units' final activations at-1 (lower graph), are moved toward the target yt-1+St at-1 (upper graph): this causes the votes of the winning cluster of accumulator units to increase (bold arrow) while other votes are not changed. Right: with a surprise St < 0, actor's votes yt-1 are moved toward the target yt-1+St at-1: this causes the votes of the winning cluster of accumulator units to decrease, while other votes are not changed

Description of the eye-arm robotic system and of the environment

The environment: The environment is constituted by the screen of a 19" CRT monitor showing images of patterns composed by 3 circles which can be red or green and have a diameter of 5 cm, while the background is black. The monitor is positioned horizontally, toward the ceiling, and will be used as working plane for the arm. The arm must reach one of the circles shown on the screen.

The system's real-camera "eye": The eye of the system is a standard digital web-cam (320×240 pixels, RGB) mounted over (and looking at) the work plane of the arm. The neural controller is connected to the webcam through Sun standard libraries: "JMF - Java Media Framework".

The simulated system's arm: The architecture is being tested both with a simulated and with a real robotic arm. The simulated arm has the same structure and dimensions of the real one, and it is used to run the 1-3 learning phases, so the networks trained on the simulated arm can then be used for the fourth learning phase, the reinforcement learning, both on the real and simulated arm. The arm is composed of three segments: upper arm (15.9 cm), lower arm (17.5 cm) and hand-like segment (9.5 cm). The arm is trained to keep the hand-like segment parallel and near to the screen, reducing the number of free dimensions to two instead of four, and avoiding so the redundancy problem.

The robotic system's arm: The figure below shows the robotic system. With respect to the use of this arm it is important to notice that after further investigations, ISTC-CNR decided to abandon the idea of using commercial robotic-arms for its tests (such as the ActiveMedia arm previously considered) for the following reasons:

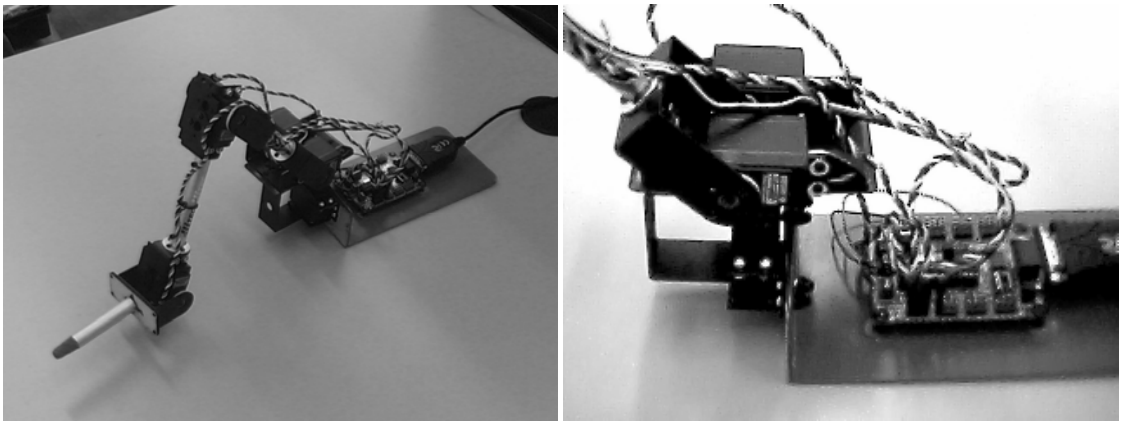
- Professional reliable robotic arms (in particular arms that do not break easily) cost too much for the project's budget (about 70 000 euros). On the other side, robotic arms that could be purchased within the project's budget (about 5 000 euros) were too fragile.
- Out-of-the-shelf robotic arm are a "black box" for research: it is not possible to easily modify the hardware as desired; moreover, repairing the hardware implies sending it back to the producer, resulting in months of lost research activity.
- Low-precision hardware was OK for the research project as it represents an interesting challenge for neural-network controllers, supposed to be robust and adaptive.

For these reasons ISTC-CNR decided to build a customized arm in its labs, that will be used to test the integrated architecture proposed here. The robotic set-up has the following features:

- The system is composed of a robotic arm and a web-cam. The robotic arm has 3 segments with 4 degrees of freedom (shoulder: 2; elbow: 1; wrist: 1) .The physical supports parts of the set-up belong to the *Erector Set* series, manufactured by *Lynxmotion* (http://www.lynxmotion.com/). The support parts for the "shoulder" of the arm were built at LARAL by customising metal blades.
- The digital servo-motors are manufactured by *Hitec*. They can be controlled in terms of desired angles. They weight 60 g each, have a torque of 12.1 kg cm, use 6 V dc., and contain a micro-controller for the regulation of velocity, power, maintenance of desired position, etc. A suitable

"programmer", HFP-10, can be used to set the parameters of the servo-motor (e.g., positions' range, rotation direction, *failsafe*, maximum speed, etc.).

- The electronic card used to control the servo-motors is an SSC32 (*Serial Servo Controller*), distrubuted by *Lynxmotion*. It is based on the micro-controller ATMEGA8-16PI, produced by *Atmel*. The card can control up to 32 servo-motors, through 3 different electricity currents. The card has a memory device, a 24LC32P EEPROM (*Electrically Erasable and Programmable* ROM, extendible to 1024 KB).
- The card can be controlled from a pc, via serial port: (a) on the basis of C++ programs, using an API based on a DLL furnished with the card; (b) using "Java Communications API", for the control of the serial port.



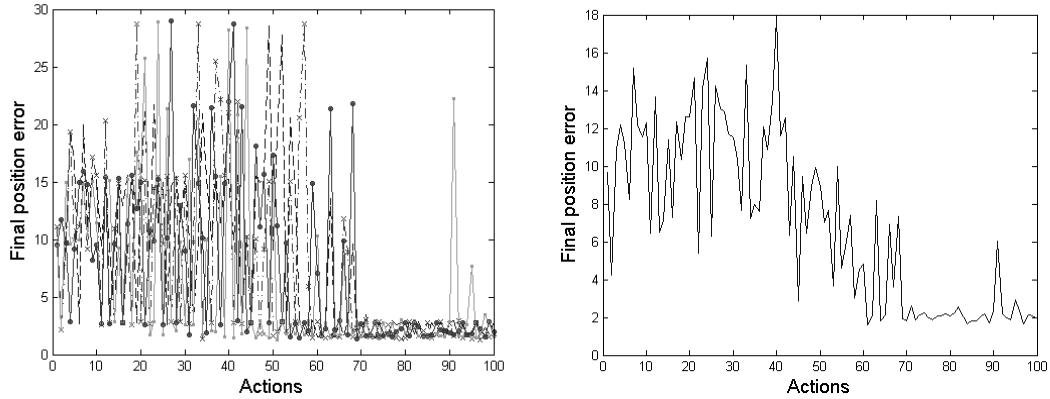
Left: robotic arm, assembled at ISTC-CNR, which is used to test the integrated architecture. Right: the base of the arm with the electronic card used to control it. A webcam, not shown in the picture, "observes" the working area of the arm from above and constitutes the "eye" of the system.

2.8.3 Experiments

In this section I'll report the experiments made over our architecture, with the use of webcam taken input and in the last experiment the real arm.

Experiment 1: Test Of Speed of Learning Phase 4 (Reinforcement Learning)

This experiment aimed to test the speed of the reinforcement learning algorithm (learning phase 4) that exploit the results of phase 3. We measured the distance between the reached position and the actual goal for the first 100 actions of 5 different simulation runs. In this test the arm was simulated whereas the input was acquired through the real webcam "watching" the stimuli screen.



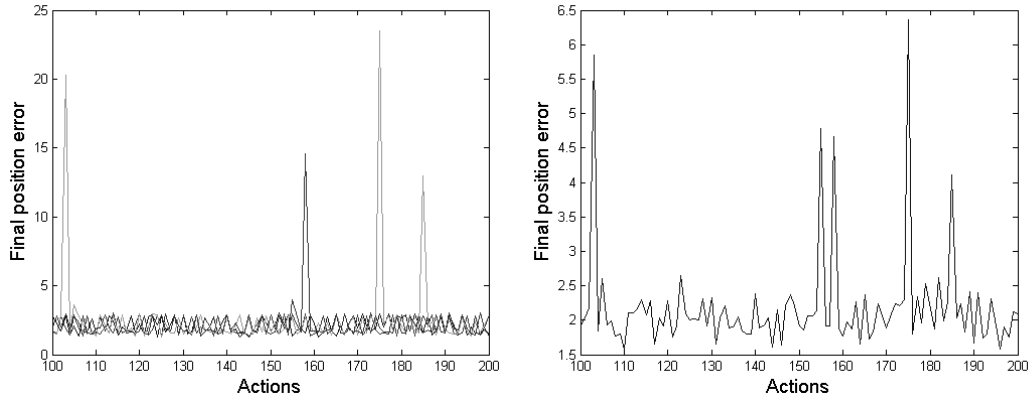
(a) (b)

Reinforcement Learning Speed Test: Final hand position error (y axis) in 100 actions (x axis). The error was measured as the Euclidean distance between the final position and the centre of the target measured

in the visual space. A data relative to 5 runs of 100 actions. B evolution of the error mean value of the five run.

After 70 actions the median error keeps its value under 5 cm, the maximum error is fewer than 3 cm after step 70 in the different runs while the correct target is chosen. This value is smaller than the dimension of the target so no more precision enhancement can be expected. The error is about the 8% of the arm extension.

Summarizing after about 70 actions the RL algorithm learns which action associating with each visual pattern. The remaining errors are caused by the 'explorative' noise and by webcam image (alignment, lights reflections and slow refresh) noise.



(a) (b)

Reinforcement Learning Accuracy: Final hand position error (y axis) in 100 actions (x axis). The error was measured as the Euclidean distance between the final position and the centre of the target measured in the visual space. A data relative to the last 100 actions of 5 runs of 200 actions. B: evolution of the error value average over the five run.

Experiment 2: Accuracy of Reaching After Learning Phase 4

This experiment aimed to test the accuracy that the reinforcement learning algorithm can achieve. The distance between the reached position and the actual goal for 100 actions after the first 100 actions for 5 simulation runs was measured. In this test the arm was simulated whereas the input was acquired through the real webcam “watching” the stimuli screen.

The medium value of error averaged over the last 100 actions and the five runs is 2.1831 cm. As shown in the figure, there are 2 runs that do not have any error, and for these runs the average value of error is 2.01cm. Two of the other three runs have one target error selection, and the last run has 3 errors that can be interpreted as some exploration behaviour.

Usually there are not large variations on the error if there are not target selection errors, as one can see from the low standard deviation of the 2 runs without target selection error: these are 0.57 and 0.64 (the runs with 1 error have 1.35 and 1.22 and the run with 3 error have a variance of 3.02cm).

Experiment 3: Performance without learning phase 3 (pre-training)

This experiment aimed at testing the performance (precision and speed) that the reinforcement learning algorithm could obtain without the third phase of learning. The distance between the reached position and the actual goal was measured for 500 actions. In this test the arm was simulated whereas the input was taken by the real webcam.

We can see from the figure below: ***A*** that the system does not seem to learn the task in 500 trials. ***B*** shows that before step 200 the error decreases in a significant way. ***C*** shows that in about half of the action the distance is low like in the experiments with the learning phase 3 while the other half has very high error. ***D*** shows a screenshot of the simulation where one can see that only 2 patterns out of 4 have been learnt, so explaining the previous result. ***E*** indicates that the precision acquired on the learnt patterns is smaller to that of experiments 1 and 2 (about 2.7 cm): this indicates that the third phase of

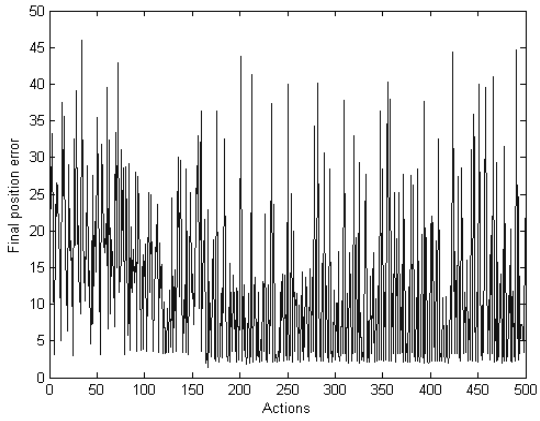
training does not produce any negative interference on the RL learning (learning phase 4). ***F*** shows that the learning of the first pattern is accomplished around action 100 and that the learning of the second pattern is accomplished around action 170, so the system is slower than when using phase 3.

One last interesting thing is that the system does not seem to be able to learn the other two patterns running for more than 300 actions after the last pattern learning. Looking at ***D*** we can suppose that this problem can be caused by a bias induced by the learning of the first 2 patterns that make the system select positions between the 2 positions relative to the two previously learnt patterns. Moreover the non-linearity of the system takes the exploration near the edge of the reachable positions. In summary the chances to find the right response for the pattern decrease for every pattern learnt.

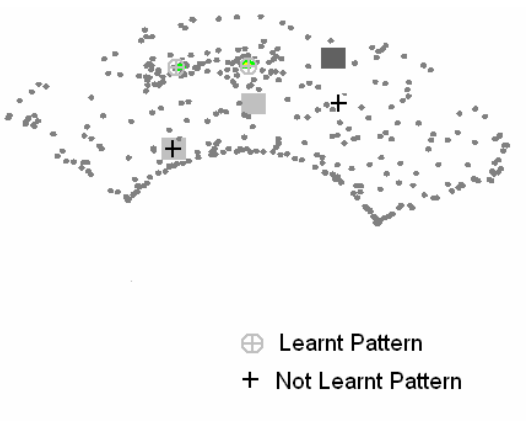
Experiment 4: Learning single patterns (Reinforcement Learning)

This experiment aimed to find some relationship between the spatial structure of the pattern and the learning speed. There are various factors that can affect learning speed with patterns that can look very similar, for example the non-linearity of the mapping and the reflections on the screen. The distance between the reached position and the actual goal was measured for 200 actions in 4 simulation the exposing the system to a single pattern for the whole simulation. In this simulation the arm was simulated while the input was taken using the real webcam.

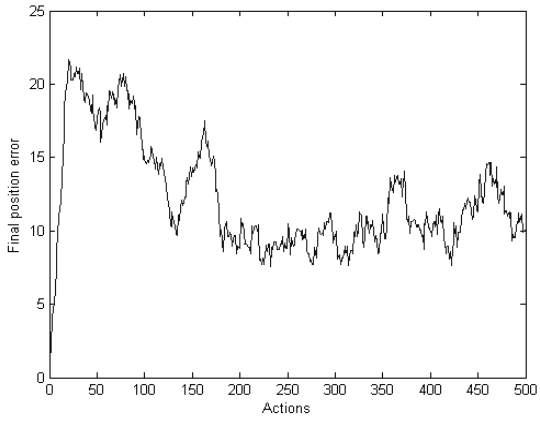
The first pattern (*Fehler! Verweisquelle konnte nicht gefunden werden.*.***A***) is learnt in very few actions (3) and the initial activations relative to the three targets seem very separated, with a cluster more active than the others, the cluster near the bottom. The second and third pattern are learnt more slowly, with about 10 actions, and their activations look less separated then that of the first pattern and without a more active cluster. The fourth pattern takes 25 actions to be learnt, and it produces an activation where cluster are very difficult to be distinguished



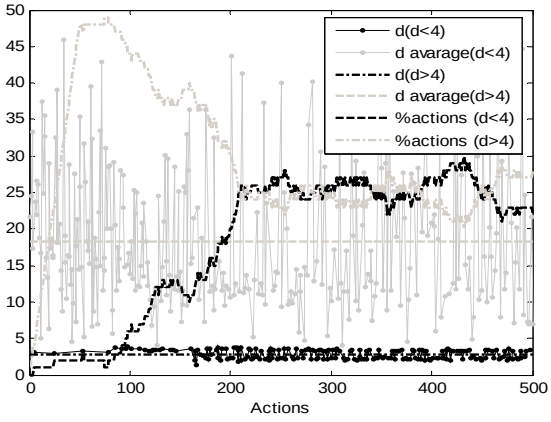
(a)



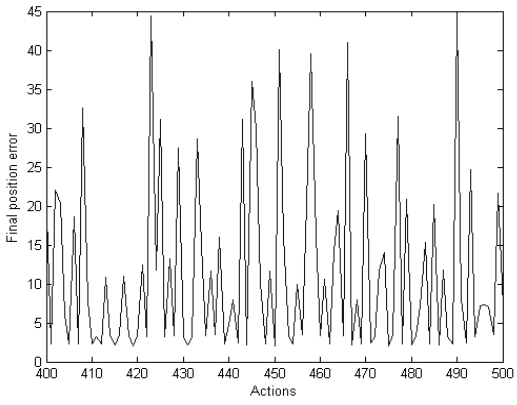
(d)



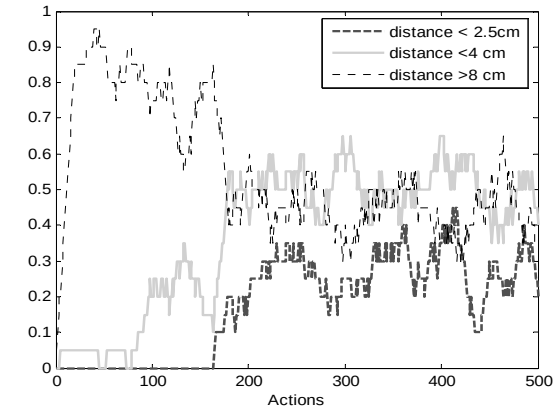
(b)



(e)

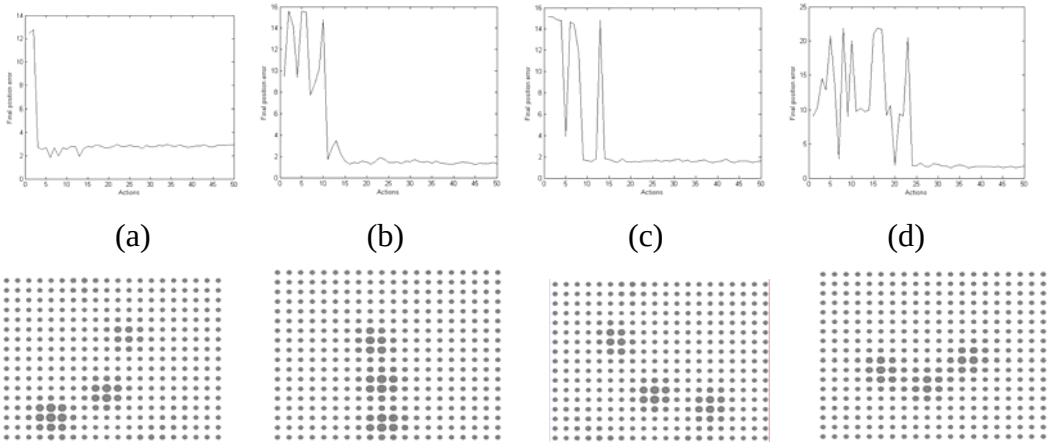


(c)



(f)

Reinforcement Learning without phase 3:(A,B,C) Final hand position error (y-axis) on actions(x-axis). The error is the Euclidean distance between the final position and the centre of the target measured in the visual space. A: evolution of the error. B: evolution of the error averaged with a moving average of 20 steps. C: evolution of the error for the last 100 actions. D image taken from the simulation, the dots are reached positions, the 3 rectangles are the pattern shown to the webcam, the crosses are the target positions for the various patterns. E diagram plots different data when the distance (d) is smaller then 4 and when it is greater then 4, the average distance d in these 2 situations and the ratio on of this two situations. F: the ratio with different thresholds distance threshold (2.5,4,8cm)



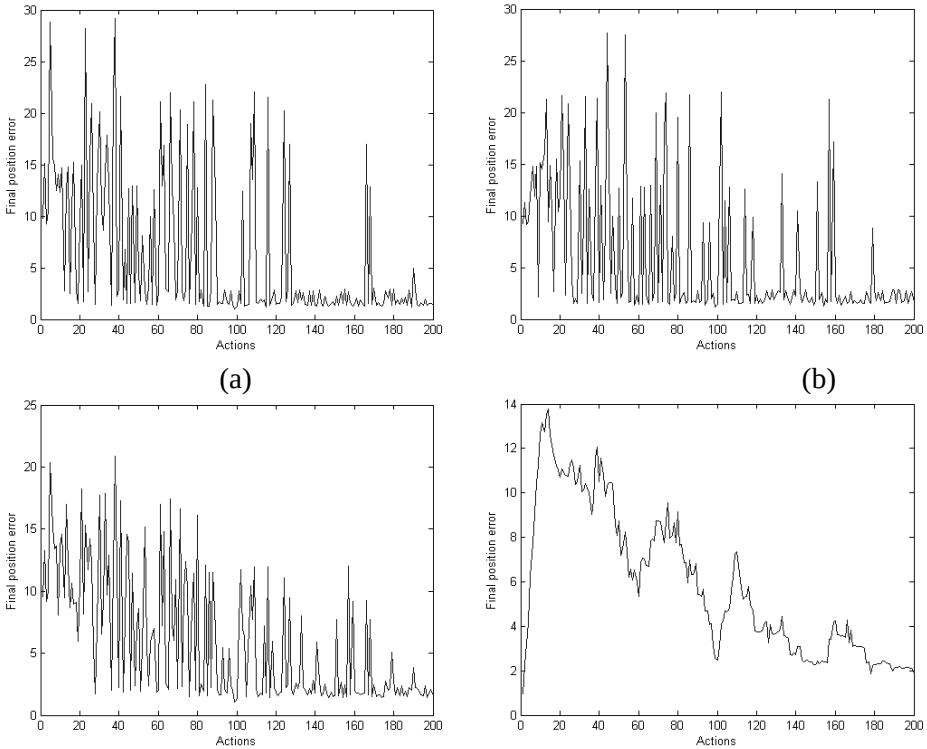
(a) (b) (c) (d)

Performance for different patterns :(ABCD)Top figure is final position error versus action index. The bottom figure is a screenshot of the activation of the votes map before training. On each column we have the two figures for the different patterns.

Experiment 5: Performance with the real arm (Reinforcement Learning)

This experiment aimed to test the influence of using the real arm with reinforcement learning algorithms. The distance between the reached position and the actual goal for the first 200 actions for 2 simulation runs was measured. The real arm can influence the performance with respect to simulation because it covers the image of patterns seen by the webcam.

The real arm has a learning time that is up to 2 time slower with respect to simulations. The system learn the 4 patterns around action 140 but there are many more errors probably due to the reflection of the arm on the screen and to the variation of the patterns that are covered by the arm. If the system could know the position of the arm, it could infer the pattern that is covered in a easier way, (cancellation effect): this might suggest possible future solution to this problem.



(a) (b)

(c) (d)

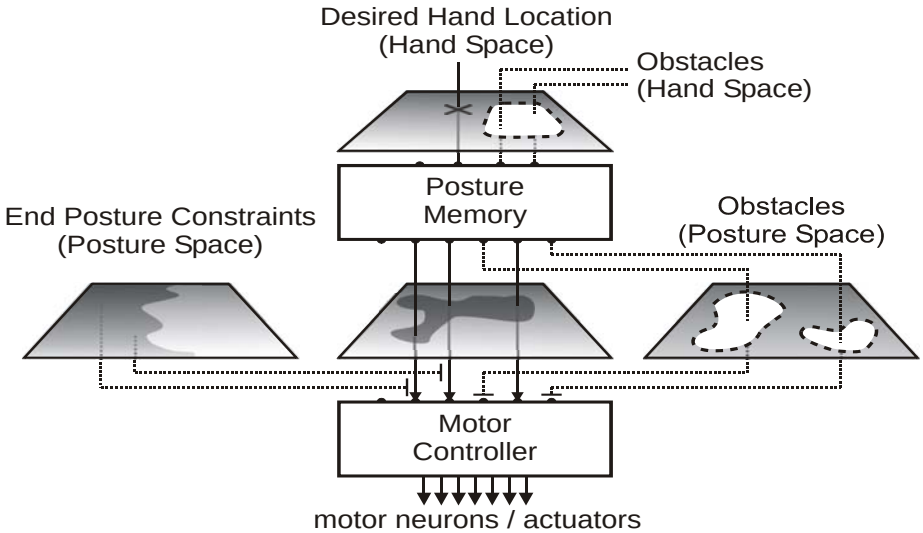
Performance with the real arm: (A,B,C,D) Final hand position error versus(y index) in 200 actions (x axis). The error is the Euclidean distance between the final position and the centre of the target measured in the visual space. Data relative 2 runs of 200 actions. A: the evolution of the error in the first run. B: evolution of the error on the second run. C: average of the 2 runs. D: evolution of the error of the 2 runs, averaged with a moving average of 10 steps.

2.9 Goal-Initiated behavior execution and control (UW)

Besides the system of ISTC-CNR that is able to select goal states effectively based on top-down motivations and bottom-up stimuli properties, UW developed a Sensorimotor Unsupervised Redundancy Resolving Architecture. (*SURE_REACH*). SURE_REACH is a modular hierarchical architecture that solves the inverse problem of generating a sequence of motor commands, which moves the hand to a desired hand location. It is divided into two modules that are trained with unsupervised Hebbian-like learning rules. SURE_REACH is currently accepted at IJCNN and also accepted for the Psychological Review journal (Herbort, Butz, in press, Butz, Herbort, Hoffmann, in press).

During learning, a posture memory (PM) addresses the inverse kinematics problem. It transforms a hand location into a set of all those arm postures that realize the respective hand location. A motor controller (MC) generates motor commands that move the arm toward the goal posture set provided by PM. Thereby MC is able to generate movements toward redundant, under-constrained goal specifications. Even more so, the postures encoded in the goal representation can be weighted if not all end postures are equally useful outcomes of the movement. MC consists of several motor command dependent body models, which encode the movements of the arm in posture space, given a certain motor command is executed.

Before a movement can be performed, MC prepares a state-to-action mapping by dynamic programming based on the learned body models. This mapping provides suitable motor commands to move a simulated arm from each possible posture toward the desired hand location and can be considered an online generated inverse model. The dynamic programming is also one of the key differences to previous models. While other models encode a single inverse model during motor learning, which is used for all tasks later on, SURE_REACH generates an individual inverse model for each task. This enables the incorporation of task-dependent constraints and optimality criteria by adjusting the model, which is used by dynamic programming, to the current task demands. For example, SURE_REACH avoids obstacles in hand space, regards novel cost functions, or controls an arm despite fixed joint angles, all without having been in either of these situations and without the necessity to relearn.



As a model for motor learning and control, SURE_REACH revealed several properties. First, during human motor learning, accuracy increases and movement times and reaction times decrease. The model can not only account for the increasingly accurate movements, but also training effects that relate to movement preparation are reflected in the model. Also movement times are reduced by training. Second, representing goals by population codes is not only in line with neurophysiological data but also with psychological findings and theories (Erlhagen & Schöner, 2002; Flash & Sejnowski, 2001). Also, the combination of the population encoded goals and the activation propagation process proved suitable to replicate human experimental data from a priming paradigm. Additionally, in contrast to many other models of motor learning and control, which can only process a single target

posture or hand location, SURE_REACH can account for more complex target representations, such as ranges of acceptable end states. Likewise, human subjects and primates are able to partially prepare movements to subsets of movement directions or distances (Bock & Arnold, 1992; Bastian, Schöner, & Riehle, 2003). Third, a priming experiment could be replicated by means of the space representation and network dynamics. Fourth, it was demonstrated that representing kinematic redundancy enables the simulation of some features of human motor control, that cannot be accounted for by kinematic models based on goal directed learning schemes (such as Baraduc et al., 2001; Ognibene et al., 2006). In humans, the final arm posture of a movement depends on the starting posture (Cruse et al., 1993; Fischer et al., 1997; Jaric et al., 1992; Soechting et al., 1995). Also in SURE_REACH, movements to the same hand position differ, depending on the starting posture.

Finally, due to the sensorimotor redundancy encoded in the motor controller, SURE_REACH is able to react quickly to novel constraints. This enables the controller to avoid obstacles and recruit alternative actions, if previously optimal actions are suddenly costly or even impossible. The successful simulation of human motor behavior due to the encoding of motor redundancy could not be accounted for by models that strive to resolve redundancy before learning.

The simulations in Herbort & Butz (in press) showed additionally that SURE_REACH can exploit the kinematic redundancy to incorporate demands of the subsequent task in its goal representation. By doing so, the subsequent movement can be carried out faster because it starts from an advantageous posture. The suitability of a posture to serve as starting point for a particular task is provided by the sensorimotor grounded distance measures in the motor controller. Similar behavior in humans has been found in reaching tasks (Fischer, Rosenbaum, Vaughan, 1997) but also in other domains like bimanual object manipulation (Weigelt, Kunde, & Prinz, 2006) or speech production (Dell, Chang, & Griffin, 1999). Additionally, the more complex movement preparation process is in line with experimental findings, which show an increase in preparation time for the initiation of the first movement of a sequence of aiming movements (Lavrysen, et al., 2003). In conclusion, the availability of redundant postures provides the flexibility to align movements to the demands of future tasks.

In conclusion, in contrast to many other models, SURE_REACH relies on an unsupervised learning scheme that connects neurons that encode different body configurations. This is very appealing from a computational and neuroscientific point of view. On the one side, body states or movement plans are likewise represented by populations of neurons in different motor areas (Georgopoulos, 1995). Theoretical considerations suggest that this form of representation is not only robust but enables advanced information processing (Knill & Pouget, 2004). On the other side, associative learning mechanisms were substantiated in motorcortical areas (Jackson, Mavoori, & Fetz, 2006). Furthermore, a big problem of error based motor learning approaches is avoided because associative learning does not require the transformation of an externally represented error signal into an error signal in motor command space.

The reported simulations clearly show that the representation of complete kinematic and sensorimotor body models enables the quick adaptation to novel tasks and optimality criteria. Whereas striving to encode redundancy may be very economic in the sense that most of what is experienced is also retrievable, it requires far more complex neural networks to encode the body models and to generate motor commands task-dependently than neural networks that encode a direct goal-to-action mapping. Hence, to apply this architecture to control a body with many more degrees of freedom, the learning mechanisms and body representations have to be refined. First, many independent low dimensional body models could be stored in a modular architecture, for example, separating arm, hand, and finger representations. Second, the now hard-wired population encoding could be improved by self-organizing maps that optimally cover the relevant work space. Third, local learning rules and sparse neural networks could replace the now fully connected neural networks.

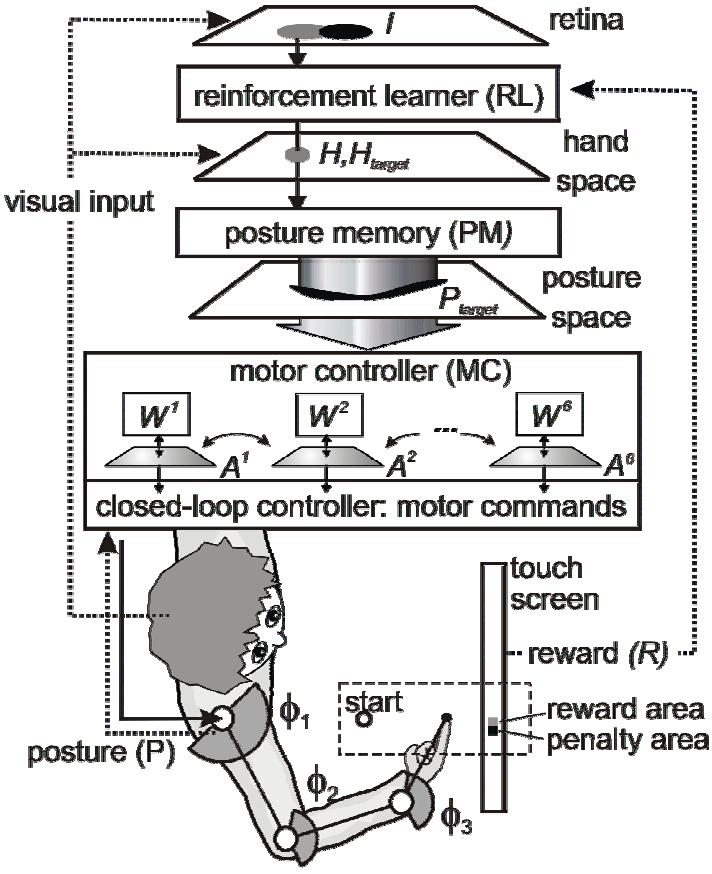
By including these enhancements we are confident that SURE_REACH will be able to control more complex and dynamic bodies. In conclusion, learning mechanisms that encode only a single goal-to-action mapping are too restricted to account for the high flexibility in human motor behavior. On the other hand, it is not yet well understood how the human brain can adapt from one second to the other to the requirements of constantly changing tasks in a constantly changing environment, let alone how this competency might be acquired. SURE_REACH makes one step forward to deepen our understanding of this intricate question and further experiments with SURE_REACH will shed additional light on the underlying computational, neural, and psychological mechanisms.

2.10 Combined Goal-initiation Architectures for Interactive Goal-directed Action Selection and Execution (ISTC-CNR and UW)

The two previous systems were recently combined into a RL_SURE_REACH (Herbot, Ognibene, Butz, Baldassarre, submitted). A general outline of the combined architecture and its capabilities is given in the following.

The paper presents a developmental neural network model of motor learning and control, called RL SURE REACH. In a childhood phase, a motor controller for goal directed reaching movements with a redundant arm develops unsupervised. In subsequent task-specific learning phases, the neural network acquires goal-modulation skills. These skills enable RL SURE REACH to master a task that was used in a psychological experiment by Trommershäuser, Maloney, and Landy (2003). This task required participants to select aim points within targets that maximize the likelihood of hitting a rewarded target and minimizes the likelihood of accidentally hitting an adjacent penalty area. The neural network acquires the necessary skills by means of a reinforcement learning based modulation of the mapping from visual representations to the target representation of the motor controller. This mechanism enables the model to closely replicate the data from the original experiment. In conclusion, the effectiveness of learned actions can be significantly enhanced by fine-tuning action selection based on the combination of information about the statistical properties of the motor system with different environmental payoff scenarios.

The figure on the right shows the four main components of the architecture. (1) A model of the human motor apparatus and the experimental setup receives arm motor commands and provides proprioception of current joint angles, visual information about the hand position and target locations in extrinsic space, and overall reward values to the neural controller. (2) A *motor controller (MC)* generates step-by-step motor commands to move the arm toward target postures (Ptarget). MC is trained by unsupervised associative learning in an initial "childhood" learning phase, during which random motor commands are executed and their effect on joint postures are encoded. Later on, this information is used, in an inverse fashion, to map from (desired) arm postures to motor commands. (3) As the task requires movements to targets represented in an extrinsic coordinate frame, a *posture memory (PM)* converts a hand target (Htarget), encoded in extrinsic coordinates, into a representation of the redundant arm postures that correspond to it. The output of PM is used as the target representation for MC. Like MC, PM develops in an unsupervised fashion during the childhood learning phase. (4) Finally, an actorcritic *reinforcement learning (RL)* mechanism [9] modulates the retinal input (I) before it is used as target representation forMC. During task specific learning phases in the simulation of the experiment, RL explores the consequences of the selection of varying target representations. Thereby it "cristalizes" on a retinal-to-target representations mapping that, given the configuration of the reward/penalty areas as well as neural and motor noise, maximizes the overall payoff.



In summary, the paper presented the RL SURE REACH architecture, which is used to model the acquisition of basic motor skills with unsupervised learning during a childhood phase and their use for the acquisition of task-specific skills with reinforcement learning in a later phase. The model was validated by successfully reproducing data obtained in a psychological experiment, in which

participants had to hit a rewarded area on a touch screen while avoiding touching penalty areas, facing various cost and position configurations. In these tests, similarly to humans, the model exhibited a remarkable capability of shifting movement endpoints within the target area taking into account the possibility of hitting the penalty areas due to motor and neural noise. Most neural-network models of motor learning and control proposed so far focus on the extraction of compact representations of sensory-to-motor mappings. In this respect, the experiments presented here show that adding reinforcement learning components to such models can enables a sensorimotor control loop to take into account the statistical properties of the motor system. This can be very important to effectively solve the location-redundancy problem and thus increase behavioral performance. Neural population codes, as used in RL SURE REACH, seem to be well suited to encode knowledge about the statistical properties of tasks and our sensorimotor systems [11], [12]. The results reported here show that this knowledge is necessary to achieve ones goals optimally accounting for sensorimotor uncertainty.

2.11 Improvements in Analogy-Based Anticipations (NBU)

NBU published several papers on goal-directed anticipatory behavior based on reasoning by analogy. AMBR forms the core of NBU's architecture. The IKAROS system (from LUND) was integrated in two of the publications in order to handle real-world perception information. Also comparisons of the behavior of the resulting architecture with psychological data were carried through. The architecture is tested in 'finding an object' scenario in several environments. The experiments were either carried through in simulation or with a real robot.

Although there are currently no direct comparisons with other architectures, it is planned to compare the performance of the AMBR/IKAROS model with the performance of a connectionist model. Hereby, the task will be likely the same used so far, that is, to find a bone hidden under one of three shapes. Every model would have to identify the three shapes and to point to the one under which the bone is hidden. After each trial the system will receive feedback about the 'correct' solution.

The bone will be hidden using one of following strategies:

1. Same property -e.g. 'The bone is always under a red shape'
2. Same relation - e.g. 'The bone is under the shape with unique color'
3. Complex - e.g. 'The bone is under the left object of two objects with same color.'

A Neural Network model could be seen as a traditional approach in such tasks. In preliminary experiments, a simple NN was trained to solve a task using strategies of type 1. It is planned to collaborate with a project partner for finding a NN that provides good results for the other two strategies. But, of course, it is expected that the AMBR/IKAROS approach will provide better results than most NNs because: (1) It can generate predictions using a small number of memorized situations (even one past episode is enough for making analogies). (2) It can 'catch' complex rules like 2, 3 (shown above). (3) It can give correct solutions even when using different hiding strategies. (4) It is context sensitive so that in different contexts different solutions can be generated.

The first paper (Petkov, et al, in press) uses the AMBR model of analogy-making as a basis, but it extends it with new agent-types and new mechanisms that allow anticipating in relation with analogical transfer. The role of selective attention on retrieval of memory episodes is tested in a series of simulations and demonstrates the context sensitivity of the AMBR model. The results of the simulations clearly demonstrated that endowing robots with analogy-based anticipatory behavior is promising and deserves further investigation.

The second paper (Petkov, Kiryazov, Grinberg, Kokinov, in press) investigates anticipation by analogy further. First, the role of selective attention is explored both with simulation data and within psychological experiment. After that, the AMBR model for analogy-making is extended with a simple anticipatory mechanism and it is demonstrated how top-down perception and analogical transfer can both be based on one and the same anticipatory mechanism. Finally, attention and action mechanisms are added to the model and AMBR was implemented in a real robot that behaves in a natural environment.

Finally, the third paper (Kiryazov, Petkov, Grinberg, Kokinov, Balkenius, in press) outlines an approach to building robots with anticipatory behaviour based on analogies with past episodes. Hereby, anticipatory mechanisms are used to make predictions about the environment and to control selective attention and top-down perception. An integrated architecture is presented that perceives the

environment, reasons about it, makes predictions and acts physically in this environment. The architecture is implemented in an AIBO robot. It successfully finds an object in a house like environment. The AMBR model of analogy-making is used as a basis, but it is extended with new mechanisms for anticipation related to analogical transfer, for top down perception, and for selective attention. The bottom up visual processing is performed by the IKAROS system for brain modelling. The paper describes the first experiments performed with the AIBO robot and demonstrates the usefulness of the analogy-based anticipation approach.

2.12 Motivations and Schemas in Interplay (ISTC-CNR)

ISTC-CNR continued its research on schema-based architectures that now include motivational drives in elaborate schemas. The interplay of the schemas is highly interesting and shows typical behavioural effects such as over-drive to hunger. The publications also compare reactive and anticipatory strategies in a predator-prey scenario. The three papers are introduced shortly in the following texts.

In Pezzulo and Calvi (2006) a schema-based agent architecture, which is inspired by an ethological model of the praying mantis, was introduced. The architecture includes an inner state, perceptual and motor schemas, several routines, a fovea, and a motor controller. The design and implementation of the architecture is described and it is used for comparing two models: the former uses reactive, stimulus-response schemas whereas the latter involves also forward models, which are used by the schemas for generating predictions. Our results show an advantage in using anticipatory components inside the schemas.

In the second paper (Pezzulo, Calvi, Castelfranchi, 2007), a layered (schema-based and deliberative) architecture was implemented and tested in the guards-and-thieves scenario, also comparing it with other strategies such as A* and BDI. DiPRA (A Distributed Practical Reasoning Architecture) implements the main principles of practical reasoning via the distributed action selection paradigm. We introduce and motivate the underlying theoretical and computational peculiarities of DiPRA and we describe its components, also providing as a case study a guards-and-thieves task.

Finally, in Pezzulo and Calvi (in press) a theoretical analysis of *schema-based design* (SBD), which is a methodology for designing autonomous agent architectures, was presented. Besides SBD; an overview of the *AKIRA Schema Language (AKSL)* is given, which permits to design schema-based architectures for anticipatory behavior experiments and simulations. Several simulations using AKSL were reviewed, highlighting the relations between pragmatic and epistemic aspects of behavior. It is shown that anticipation was crucial in realizing several functionalities with AKSL, such as selecting actions, orienting attention, and categorizing and grounding declarative knowledge.

2.13 Anticipatory Coordination (ISTC-CNR)

The work package efforts extended even further into the social aspects of anticipatory agents. First, Piunti, Castelfranchi, and Falcone (2007) published a paper on anticipatory coordination through action observation and behaviour adaptation.

The paper proposes a computational multi agent system (MAS) where agents are built to exploit environment traces, "mind reading" and plan recognition capabilities in order to anticipate other agents' mental states and therefore adapt their behavior for an anticipatory social interaction.

An agent exploits such mind reading capabilities by observing other agent during their practical behavior and by including "expectations" about their future behaviour in its reasoning process. On the basis of a recognized pattern (i.e. a plan) an observer may predict what sequence of actions will be performed by an observed agent. By so doing, observer agent is leaning to be anticipatory regards other ones in competitive or cooperative terms. Once an expectation about an observed agent is given, the agent has two alternatives: (1) Try to modify its own behavior to exploit or avoid the outcomes of actions performed by others. (2) try to influence the behaviour of the observed agents, in order to help or prevent their goals.

An experiment and a case study are given, showing all the phases of the anticipatory interaction between agents, from recognition of behavior to intention reconsideration and behaviour adaptation. The work does only present a computational model in its early release, without either performance analysis or evaluation of MindRACES metrics of interest.

2.14 Backward vs. Forward-oriented Decision Making in the Iterated Prisoner's Dilemma: A Comparison between Two Connectionist Models (NBU)

Lalev and Grinberg (in press) study two recurrent neural network architectures playing the iterated prisoner's dilemma. Both models are based on common recurrent network architecture. While the first model used backward-oriented reinforcement learning methods, the second network basis its movement decisions on generated predictions about future games. Thus, both models involve predictions of the opponent's move and of the expected payoff and have an in-built autoassociator in their architecture aimed at a more efficient payoff matrix representation. However, only the latter network anticipates the actual behavior of the opponent player. The role of the models' building blocks and mechanisms is investigated and discussed and finally, comparisons with experiments with human participants are presented. The results suggest that human players use anticipatory capabilities to guide their decision process within the game. As with actual human participants, the cooperation rate of the latter network depended on a so-called cooperation index, which quantifies the likelihood that the opponent player cooperates. Thus, the results suggest that anticipatory connections are mandatory for efficient human-like network interaction within the iterated prisoner's dilemma game.

2.15 An Experimental Study of Anticipation in Simple Robot Navigation (LUND)

Finally, Johansson and Balkenius (in press) study the benefits of anticipating the behavior of another robot agent. They placed two real robots in differently complex arenas with the task of switching places with each other. The results show that in very simple environments without obstacles, a goal-directed behavioral strategy without any consideration of the opponent player, except for a reactive hard-coded obstacle avoidance mechanism, yielded the most efficient behavior. However, in more complex environments, in which robot interference is inevitable and harder to resolve, anticipatory mechanisms yielded the fastest behavior. In this case, the anticipatory mechanism predicted the behavior of the opponent robot and resolved possible trajectory conflicts online. Thus, it is shown that higher complex environments can make more complex, cooperative, anticipatory mechanisms beneficial. In very simple interactive environments, on the other hand, ignorance of the opponent or cooperative player can also be more effective, since no expensive contemplations or communicative interactions are necessary.

Thus, the paper essentially presents an experimental study using two robots. In the experiment, the robots navigated through an area with or without obstacles and had the goal to shift places with each other. Four different approaches (random, reactive, planning, anticipation) were used during the experiment and the times to accomplish the task were compared. The results indicate that the ability to anticipate the behavior of the other robot can be advantageous. However, the results also clearly show that anticipatory and planned behavior is not always better than a purely reactive strategy.

3 Conclusions

As deliverable 4.1 had, this document shows that there are many types and challenges for anticipatory behavior systems. Solutions are never straight forward and it is often hard to estimate if anticipations will be useful at all in the applied context.

In the introduction, we had proposed the further investigation of

- the XCS architecture, which was accomplished by the XCSF advancements;
- the artificial immune system architecture (AIS), which is now applied to an AIBO robot control task;
- inverse gradient methods and reinforcement learning techniques, which were applied to predict deep memory POMDP problems as well as for robot navigation;
- inverse model-based systems, which are now being combined for target selection and control problems for redundant arm control and efficient target location selection
- context-based systems, which are now included in several top-down bottom up interactions including the SURE_REACH architecture but also the analogy-making architectures.

- analogy-based systems, which was accomplished in the AMBR architecture, its combination with IKAROS, and the successful application to the AIBO platform.
- Finally, also recurrent neural network approaches were further investigated proposing a new testbed and evaluating Elman networks as an exemplary RNN.

Additionally, deliverable 4.1 pointed out that different anticipatory capabilities should be investigated further. These were tested and extended in several forms:

- Inverse modeling capabilities were enhanced within two separate arm control architectures (developed by UW and ISTC-CNR) leading to the combination into RL_SURE_REACH.
- Behavioral adjustments due to unexpected sensory inputs are accomplished in the schema architectures investigated and enhanced, since schemas depend on their accuracy to be active and also can trigger surprise effects.
- Task-dependent planning mechanisms were investigated in the robot switching places task as well as in a sense in the analogy making task.
- Motivational mechanisms were coupled with behavioral decision making and control in the schema architecture framework.
- Epistemic actions were present in the schema architecture but also in the analogy making architectures.
- Besides these accomplishments, also the social components of anticipatory behavior for goal-directed behavior and learning were studied and advanced.

Thus, we believe that we can conclude that the work in workpackage 4 progressed nicely. Collaborations are springing off, which will be further reported on in workpackage 6 and will continue until the end of the project. The main systems of the partners were continuously enhanced and improved. Moreover, several investigations have shown that novel systems needed to be created to deal with the control problem of redundant plants (realized in the SURE_REACH architecture) and also to improve social interactions. For further details on the accomplished aspects of this work, the papers, mentioned in the different sections, are available on the MindRACES webpage (www.mindraces.org).

References

Baillie, J.-C. (2005). URBI: Towards a Universal Robotic Low-Level Programming Language. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS05*.

Bakker, B., Zhumatiy, V., Gruener, G., & Schmidhuber, J. (2006). Quasi- Online Reinforcement Learning for Robots. ICRA 2006.

Baraduc, P., Guigon, E., & Burnod, Y. (2001). Recoding arm position to learn visuomotor transformation. Cerebral Cortex, 11, 906-917.

Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, 22, 577–600.

Bastian, A., Schöner, G., & Riehle, A. (2003). Preshaping and continuous evolution of motor cortical representations during movement preparation. *European Journal of Neuroscience*, 18, 2047-1058.

Bock, O., & Arnold, K. (1992). Motor control prior to movement onset: preparatory mechanisms for pointing at visual targets. *Experimental Brain Research*, 90, 209-216.

Brooks, R. A. (1991). Intelligence without representation. *Articial Intelligence*, 47 (1-3), 139-159.

Butz, M. V., & Hoffmann, J. (2002). Anticipations control behavior: Animal behavior in an anticipatory learning classifier system. *Adaptive Behavior*, 10(2):75–96.

Butz, M. V. (2002). *Anticipatory learning classifier systems*. Kluwer Academic Publishers, Boston, MA.

Butz, M. V., Goldberg, D. E., & Tharakunnel, K. (2003). Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11:239–277.

Butz, M. V., Sigaud, O., & Gérard, P. (2003). Internal models and anticipations in adaptive learning systems. In M. V. Butz, O. Sigaud, and P. Gérard, editors, *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems*, 86–109. Springer-Verlag, Berlin.

Butz, M. V. (2006) *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*. Studies in Fuzziness and Soft Computing Series, Springer Verlag, Berlin.

Butz, M. V., Goldberg, D.E., Lanzi, P.L., & Sastry, K. (2007). Problem Solution Sustenance in XCS: Markov Chain Analysis of Niche Support Distributions and Consequent Computational Complexity. *Genetic Programming and Evolvable Machines*, 5 - 37.

Butz, M. V., Herbort, O., & Hoffmann, J. (in press) Exploiting Redundancy for Flexible Behavior: Unsupervised Learning of a Modular Sensorimotor Control Architecture. *Psychological Review*.

Butz, M. V., Lanzi, P. L., Wilson, S. W. (2006). Hyper-ellipsoidal conditions in XCS: Rotation, linear approximation, and solution structure. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*. 1457-1464.

Butz, M. V., Lanzi, P. L., & Wilson, S. W. (in press) Function Approximation with XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction. *IEEE Transactions on Evolutionary Computation*.

Butz, M. V., & Pelikan, M. (2006). Studying XCS/BOA learning in Boolean functions: Structure encoding and random boolean functions. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*. 1449-1456.

Butz, M. V., Pelikan, M., Llorà, X., & Goldberg, D. E. (2006) Automated global structure extraction for effective local building block processing in XCS. *Evolutionary Computation*, 14, 345-380.

Carey, R., Bell, G., & Marrin, C. (1997) ISO/IEC 14772--1:1997 Virtual Reality Modeling Language (VRML97). http://www.web3d.org/x3d/specifications/vrml/

Castelfranchi, C. (2005). Mind as an anticipatory device: For a theory of expectations. In *BVAI 2005*, pages 258–276.

Cisek, P., & Kalaska, J. (2005). Neural correlates of reaching decisions in dorsal premotor cortex: specification of multiple direction choices and final selection of action. *Neuron*, 45, 801-814.

Cruse, H., & Steinkühler, U. (1993). Solution of the direct and inverse kinematic problems Exploiting Redundancy 68 by a common algorithm based on the mean of multiple computations. Biological Cybernetics, 69, 341-351.

Dell, G. S., Chang, F., & Griffin, Z. M. (1999). Connectionist models of language production: Lexical access and grammatical encoding, *Cognitive Science*, vol. 23, no. 4, pp. 123–147.

Drescher, G. L. (1991). *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, 1991.

Erlhagen, W., & Schöoner, G. (2002). Dynamic field theory of movement preparation. *Psychological Review*, 109 (3), 545–572.

Farmer, J. D., Packard, N., & Perelson, A. S. (1986). The immune system, adaption and machine learning, Phisica D, 22 187-204.

Fischer, M. H., Rosenbaum, D. A., & Vaughan, J. (1997). Speed and sequential effects in reaching, *Journal of Experimental Psychology: Human Perception and Performance*, vol. 23, no. 2, pp. 404–428.

Flash, T., & Sejnowski, T. J. (2001). Computational approaches to motor control. *Current Opinion in Neurobiology*, 11, 655-662.

Gallese, V. (2000). The inner sense of action: Agency and motor representations. Journal of Consciousness Studies, 7:23–40.

Georgopoulos, A. P. (1995). Current issues in directional motor control, *Trends in Neuroscience*, vol. 18, no. 11, pp. 506–510.

Grush; R. (2004). The emulation theory of representation: Motor control, imagery, and perception. Behavioral and Brain Sciences, 27(3):377–96.

Hart, E. & Timmis, J. (2005). Application Areas of AIS: *The Past, The Present and The Future*. In Icaris 2005, 3627, 483-497. Berlin Heidelberg: Springer-Verlag.

Herbart, J. (1825), Psychologie als Wissenschaft neu gegründet auf Erfahrung, Metaphysik und Mathematik. Zweiter, analytischer Teil. August Wilhem Unzer, Königsberg, Germany, 1825.

Herbort, O. & Butz, M.V. (in press). Encoding complete body models enables task dependent optimal behavior. *International Joint Conference on Neural Networks*.

Herbort, O:, Ognibene, D., Butz, M. V., & Baldassare, G. (submitted). Learning to Select Targets within Targets in Reaching Tasks, *ICDL 2007*.

Hesslow, G. (2002). Conscious thought as simulation of behavior and perception. *Trends in Cognitive Sciences*, 6:242–247.

Hoffmann, J. (2003). Anticipatory behavioral control. In M. V. Butz, O. Sigaud, and P. Gerard, editors, *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems*, pages 44–65. Springer-Verlag, Berlin Heidelberg, 2003.

Hoffmann, J. (1993). Vorhersage und Erkenntnis: Die Funktion von Antizipationen in der menschlichen Verhaltenssteuerung und Wahrnehmung [Anticipation and cognition: The function of anticipations in human behavioral control and perception]. Hogrefe, Göttingen, Germany.

Houk, J.C., Davis, J.L., Beiser, D.G. (eds.): Models of Information Processing in the Basal Ganglia. MIT Press, Cambridge MA (1995)

Jackson, A., Mavoori, J., & Fetz, E. E. (2006). Long-term motor cortex plasticity induced by an electronic neural implant, *Nature*, 444, 56–60, 2006.

James, W. (1890). The Principles of Psychology. Dover Publications, New York.

Jaric, S., Corcos, D. M., & Latash, M. L. (1992). Effects of practice on final position reproduction. *Experimental Brain Research*, 91, 129 -134.

Jerne, N. K. (1974). Towards a Network Theory of the Immune System. Annals of Immunology, 125, 373–389.

Johansson, B., & Balkenius, C. (2007, in press). An Experimental Study of Anticipation in Simple Robot Navigation. In Butz, M.V.; Sigaud, O.; Pezzulo, G. & Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520*, Springer-Verlag.

Kandel, E.R., Schwartz, J.H., Jessell, T.M. (2000). Principles of Neural Science. McGraw-Hill, New York.

Kiryazov, K., Petkov, G., Grinberg, M., Kokinov, B., Balkenius, C. (2007, in press). The Interplay of Analogy-Making with Active Vision and Motor Control in Anticipatory Robots. In Butz, M.V.; Sigaud, O.; Pezzulo, G. & Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520*, Springer-Verlag.

Knill, D. C. & Pouget, A. (2004). The bayesian brain: The role of uncertainty in neural coding and computation, *Trends in Neurosciences*, 27, 712–719.

Kohonen, T. (2001). Self-Organizing Maps. 3rd edn. Springer-Verlag, Berlin.

Körding, K. P. and Wolpert, D. M.(2004). Bayesian integration in sensorimotor learning, *Nature*, 427, 244–247.

Kuperstein, M. (1988). A Neural Model of Adaptive Hand-Eye Coordination for Single Postures. *Science*, 239, 1308-1311.

Lalev, E., & Grinberg, M. (2007, in press). Backward vs. Forward-oriented Decision Making in the Iterated Prisoner's Dilemma: A Comparison between Two Connectionist Models. In Butz, M.V.; Sigaud, O.; Pezzulo, G. & Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520*, Springer-Verlag.

Lavrysen, A., Helsen, W. F., Tremblay, L., Elliott, D., Adam, J. J., Feys, P., &. Buekers, M. J. (2003). "The control of sequential aiming movements: The influence of practice and manual asymmetries on the one-target advantage," *Cortex*, 39, 307–325.

Martinetz, T. M., Berkovitsch, S. G., & Schulten, K. J. (1993). "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4, 558-569.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1 (1), 39–42.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). Plans and the Structure of Behavior. Holt, Rinehart and Winston, New York.

Mottaghi, R., & Vaughan, R. T. (2006). An Integrated Particle Filter \& Potential Field Method for Cooperative Robot Target Tracking, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

O'Regan, J: and Noe, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5), 883–917.

Ognibene D., Mannella F., Pezzulo G., Baldassarre G. (2006). Integrating reinforcement-learning, accumulator models, and motor-primitives to study action selection and reaching in monkeys. In Fum D., Del Missier F., Stocco A. (eds.), *Proceedings of the Seventh International Conference on Cognitive Modeling (ICCM2006)*, 214-219. Trieste, Edizioni Goliardiche.

Ognibene, D., Rega, A., & Baldassarre, G. (2006). A Model of Reaching that Integrates Reinforcement Learning and Population Encoding of Postures. *From Animals to Animats 9: Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior* (SAB-2006), 381-393.

Petkov, G., Kiryazov, K., Grinberg, M., & Kokinov, B. (2007, in press) Modeling Top-Down Perception and Analogical Transfer with Single Anticipatory Mechanism. In: *Proceedings of the Second European Cognitive Science Conference, Greece*.

Petkov, G., Naydenov, Ch., Grinberg, M., Kokinov, B. (2006, in press). Building Robots with Analogy-Based Anticipation. In: *Proceedings of the KI 2006, 29th German Conference on Artificial Intelligence*, Bremen, in press.

Pezzulo G., Baldassarre G., Butz M. V., Castelfranchi C., Hoffmann J. (2006). An analysis of the ideomotor principle and TOTE. In Butz M. V., Sigaud O., Pezzulo G., Baldassarre G. (eds.), *Proceeding of the Third Workshop on Anticipatory Behaviour in Adaptive Learning Systems (ABiALS 2006)*. Rome, Istituto di Scienze e Tecnologie della Cognizione, Consiglio Nazionale delle Ricerche (ISTC-CNR).

Pezzulo, G. & Calvi, G. (2006). A Schema Based Model of the Praying Mantis LNAI in Nolfi, S.; Baldassarre, G.; Calabretta, R.; Hallam, J.; Marocco, D.; Miglino, O.; Meyer, J. & Parisi, D. (ed.), *From animals to animats 9: Proceedings of the Ninth International Conference on Simulation of Adaptive Behaviour, Springer Verlag, LNAI 4095*, 211-223

Pezzulo, G. & Calvi, G. (2007, in press). Schema-based design and the AKIRA Schema Language: An Overview. In Butz, M.V.; Sigaud, O.; Pezzulo, G. & Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520*, Springer-Verlag.

Pezzulo, G. and Calvi, G. (2006). A schema based model of the praying mantis. In S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, O. Miglino, J.-A. Meyer, and D. Parisi, editors, *From animals to animats 9: Proceedings of the Ninth International Conference on Simulation of Adaptive Behavior, volume LNAI 4095*, 211–223, Berlin, Germany, Springer Verlag.

Pezzulo, G., Calvi, G., Ognibene, D., & Lalia, D. (2005). Fuzzy-based schema mechanisms in akira. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce* Vol-2, 146–152, Washington, DC, USA, IEEE Computer Society.

Pezzulo, G.; Baldassarre, G.; Butz, M. V.; Castelfranchi, C. & Hoffmann, J. (2007, in press). From Goals to Actions and Vice-versa: The Ideomotor Principle, TOTE, and Actual System Implementations. In Butz, M.V.; Sigaud, O.; Pezzulo, G. & Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520*, Springer-Verlag.

Pezzulo, G.; Calvi, G. & Castelfranchi, C. (2007). DiPRA: Distributed Practical Reasoning Architecture *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007, 1458-1464.

Piunti, M., Castelfranchi, C., & Falcone, R. (2007). Anticipatory coordination through action observation and behavior adaptation. *Proc. of AISB'07 - Artificial and Ambient Intelligence - Mindful Environments*.

Pollack, M. E. Plans as complex mental attitudes (1990). In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, Intentions in Communication, pages 77–103. MIT Press, Cambridge, MA, 1990.

Potts, D. (2004). Incremental learning of linear model trees, Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004), 663–670.

Pouget A., Lathaam P. E.: Population Codes (2003). In Arbib, M. A. (ed.): *The Hand-book of Brain Theory and Neural Networks*. MIT Press, Cambridge MA, 893-897.

Pouget, A., Dyan, T., & Zemel, R. (2000). Information processing with populaion codes," *Nature Reviews Neuroscience*, 1, 125–132.

Prinz, W. (2005). An ideomotor approach to imitation. In S. Hurley and N. Chater, (eds.), *Perspectives on imitation: From neuroscience to social science*, 1, 141–156. MIT Press, Cambridge, MA.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge MA.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87.

Rizzolatti, G., Fadiga, L., Gallese, V., & Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3, 1996.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205.

Schaal, S. & Atkeson, C.G. (1998). Constructive incremental learning from only local information, *Neural Computation*, 1998, 2047-2084

Schall, J. D. (2001). Neural Basis of Deciding, Choosing and Acting. *Nature Reviews Neuroscience* 2 (2001) 33-42.

Schulz, D., Burgard, W., Fox , D., & Cremers, D. (2001). Tracking multiple moving objects with a mobile robot using particle filters and statistical data association, in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1665-1670.

Shadmehr, R., Wise, S. (2005). *The Computational Neurobiology of Reaching and Pointing*. MIT Press, Cambridge MA.

Soechting, J. F., Buneo, C. A., Herrmann, U., & Flanders, M. (1995). Moving effortlessly in three dimensions: Does donders' law apply to arm movement? *Journal of Neuroscience*, vol. 15, pp. 6271–6280.

Surmann, H., & Pervoelz, K. (2003). KURT3D Mobile Robotic Platform.

Sutton, R. S., Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge MA.

Thrun, S., Burgard, W., & Fox, D. (2000). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, MIT Press.

Trommershäuser, J., Maloney, L. T., & Landy, M. S. (2003). Statistical decision theory and trade-offs in the control of motor response, *Spatial Vision*, 16, 255–275.

Ulrich, I., & Borenstein, J. (1998). VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. *IEEE Int. Conf. on Robotics and Automation*, 1572—1577.

Usher, M., McClelland, J.L. (2001). On the Time Course of Perceptual Choice: The Leaky Competing Accumulator Model. *Psychological Review*, 108, 550-592.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Weigelt, M., Kunde, W., & Prinz, W. (2006). End-state comfort in bimanual object manipulation, *Experimental Psychology*, 53, 143–148, 2006.

Widrow, B., Hoff, M.E.: *Adaptive Switching Circuits* (1960). IRE WESCON Convention Record, Part 4, 96-104.

Wierstra, D. Förster, A., & Schmidhuber J. (under review) Solving Deep Memory POMDPs with Recurrent Policy Gradients.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8, 229-256.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11, 1317–1329.

Yu, Yihua and Cheng, Q. (2006). Particle filters for maneuvering target tracking problem, *Signal Process*. 86, 195-203.

Zappacosta, S.; Nolfi, S. & Baldassarre, G. (2007, in press). A Testbed for Neural-Network Models Capable of Integrating Information in Time Anticipatory Behavior. In Butz, M.V.; Sigaud, O.; Pezzulo, G. & Baldassarre, G. (eds.), *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior, LNAI 4520*, Springer-Verlag.

Zhumatiy, V., Gomez, F., Hutter, M., & Schmidhuber, J. (2006). Metric State Space Reinforcement Learning for a Vision-Capable Mobile Robot, Proc. of the Int'l Conf. on Intelligent Autonomous Systems, IAS-06, Tokyo, 2006