FP6-511931

Mind RACES

from Reactive to Anticipatory Cognitive Embodied Systems

DELIVERABLE D8 (D2.2)

Scenario Design and Implementation

Due date of deliverable:
30 / 09 / 2005

Actual submission date:
11/ 11 / 2005

Start date of project:
01 / 10 / 2004

Duration:
36 month

Organization name of lead contractor for this deliverable:
ISTC-CNR

Revision:
8

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**X**
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)	

Document identifier:	**DELIVERABLE_WP2_N_2**
Date:	**11/11/2005**
Work package:	**WP2**
Partner(s):	*IDSIA, IST, ISTC-CNR, LUCS, NBU, NOZE, OFAI, UW-COGSCI*
Lead Partner:	*ISTC-CNR*
Document status:	*Approved*
Deliverable identifier:	**WP2_D2.2**

Delivery Slip				
	Name	**Partner**	**Date**	**Signature**
From	LUCA TUMMOLINI	ISTC-CNR		
Verified	RINO FALCONE	ISTC-CNR		
Approved by	RINO FALCONE	ISTC-CNR		

Files

Software Products	**User files**
MS Word™	DELIVERABLE_WP2_N_2.DOC

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
Name:	Rino Falcone
Address:	CNR-ISTC via S. Martino della Battaglia,44 00185 Rome ITALY
Phone:	+39 06 44 595 253
Fax:	Fax: +39 06 44 595 243
Mobile:	
E-mail	rino.falcone@istc.cnr.it

TABLE OF CONTENTS

PART 1 – Management Overview

1 Document Control

This document is a co-production of all the partners mentioned above. After the individuation of the MindRACES scenarios as reported in D2.1, all the partners have started the implementation phase of the project and discussed together the results, as far as scenarios implementation is concerned, during the third project meeting. On the basis of the discussion, they have edited the contributions that are reported below.

2 Executive Summary

The objective of this deliverable is to report the activity done by the consortium to *design and implement the three selected scenarios, the tasks and the environments* that will be used in the next phases of the project. The output of this deliverable is used to develop, evaluate and test the enhanced architectures and robots whose results will be reported in D3.2, D4.2 and D5.2.

3 Terminology

The following table summarizes the working definitions used throughout the document.

Robot:	A real or simulated agent having specific sensors (e.g. camera, infrared/ultrasound sensors), actuators (e.g. two wheels, a three-segment arm), and a body (e.g. a cylinder, three rigid segments).
Mechanism:	Specific architecture and algorithms (=structure + functioning) of a model/controller.
Environment:	A particular real or simulated arena with specific features (i.e. dimensions, walls, type of "terrain"), containing particular objects (i.e. balls, boxes, doors, lights), and containing robots with specific features (i.e. sensors, actuators and bodies).
Scenario:	A set of tasks that share a common portion of an environment.
Task:	The specific goal one robot or group of robots have to accomplish in a scenario.

Table 1 Terminology adopted in the document.

PART 2 – Deliverable Content

1 IDSIA

FINDING AND LOOKING FOR

- ***Finding a specific object (Game Room)***

The purpose of this task is to find a specific object in the environment (e.g. a red cube). The degree of detail in the description must be sufficient to define unambiguously a single object, not a class of similar ones. For example "red cube" is to be used in the case when there is a single red cube, and "big red cube" if there are several red cubes with different sizes and only one of them is big.

- ***Finding members of a class of objects by class description (Game Room)***

The purpose of this task is to find any object matching some general or partial description (for example "find a cube" or "find a red object"). As in the previous case, prediction or anticipation can be based on previous experience, recurring spatial relations, etc.

IDSIA intention is to let an agent learn to find objects in a natural environment by controlling an artificial fovea. In what follows the implementation of the fovea for a real omnidirectional camera and some simulated cameras are described. The client/server framework for a real robot for easy behavior development is also presented. Eventually, the implementation of a 3D simulation of a robot in a real environment is illustrated.

1.1 IDSIA scenario in detail

The agent is an autonomous wheeled robot with a fixed camera. A process on the robot simulates a movable fovea centralis, with higher resolution in the center and coarser resolution towards the borders instead of the raw camera image. This reduces the huge data from a real camera. The fovea has attention-shifting actions such as "turn sensor right by 10 degrees". Actions for the robot are abstract driving commands too. The concept of using a fovea to reduce huge data is based on Schmidhuber and Huber (1991). They trained a feed-forward neural network to center and orientate the fovea on presented objects.

The environment is a standard office room or a prepared robot lab. Several objects cooccur frequently or are semantically related, e.g. a table, a bottle and a cork. Arbitrary degrees of difficulty are possible through complex visual scenes, partial observability, partial occlusions etc. IDSIA has also implemented some simulations for different kinds of environments for the first evaluation of learning experiments.

The robot has to find a target object in the room through active perception by producing a sequence of saccades or other movements until the target is centered in the visual field. The robot has to spot the target object as quickly as possible in the environment.

IDSIA exploits the camera image as the sole sensor of the robot. There is no intention to employ structures for explicit knowledge representation as long as it is possible to solve the tasks without them. Other preprocessing operations like edge detection and optical flow computations will only

be explicitly implemented, if it is inevitable. This is a major difference to other successful vision based robot control learning tasks. For example, LeCun et al. (2004) have built an autonomous robot that avoids hitting obstacles by the use of a multi-layered feed-forward network. However, the network was predefined to learn convolutions, e.g. for edge detection.

This section is organized as follows. First the robot and the real environments for our experiments are described. Then the transformation process of converting the omnidirectional camera image to the fovea sensor input is illustrated. The next sub-section deals with the software framework of the robot. Afterwards, the different simulated environments are presented in detail. Finally, first results and the potential modifications to the robot, the tasks, and the environments are discussed.

1.2 IDSIA Environments

IDSIA has adopted a fully autonomous Robertino robot (see Figure 1). The cylindrical robot has a diameter of 40 cm and a height of 43 cm; its weight is 6.5 kg. It is equipped with a holonomic three wheeled drive, and has a PC-103 (industry standard) with a 500MHz Intel Mobile-Pentium II processor on-board and communicates through WLAN (IEEE 802.11a). Its sensors consist mainly of an omnidirectional FireWire camera, which is used to simulate the fovea. Other sensors will not be exploited. The actuators are the three wheels and the simulated fovea, which are controlled by abstract commands for direction and velocity. A controller on the robot calculates the wheel velocities with respect to the assigned commands.

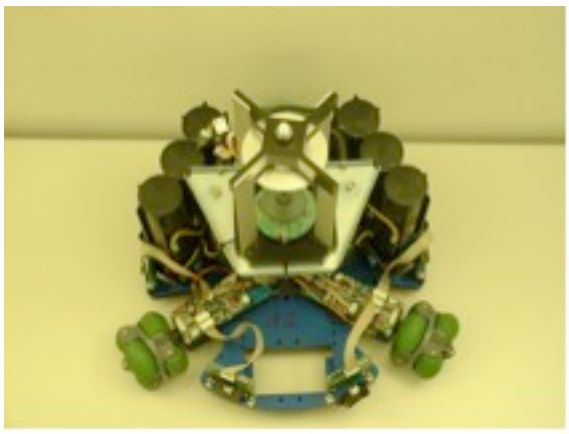


Figure 1 The Robertino robot developed by the TU München (www.openrobertino.org).

Three different kinds of real environments are used to improve the learning complexity of the physical robot step by step. The environment for the first task is a whiteboard with coloured shapes (see Figure 2). The size of the board is 150 cm x 120 cm; the shapes have diameters from 10 cm to 30 cm. The robot is placed in front of the board. It can only move its simulated fovea and cannot move around with its wheels. The task is to find a specific object, which is in general not seen at first view. The advantage of this setup is that the fovea movement is extremely fast, because no physical motion is necessary. The learning time is short in contrast to a real fovea or a real moving robot.

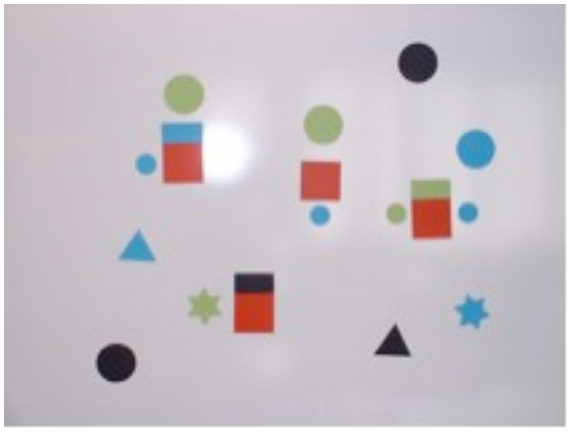


Figure 2 The whiteboard-world. Shapes of different colors are mounted on a whiteboard.

The second environment is an empty square room with a size of 550 cm x 280 cm (see Figure 3). The room is uniformly illuminated, has a grey carpet and white or grey walls. It has no windows and natural light sources; bright shining spots are eliminated. Some coloured boxes – in the size of the robot – can be arranged on the test field.

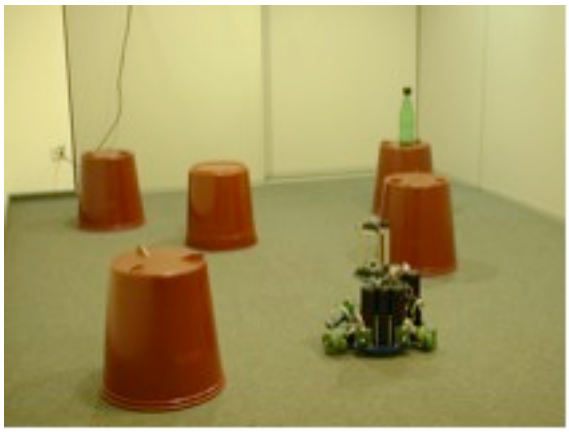


Figure 3 The small robot lab. Colored boxes are arranged on the floor. The robot must move around some obstacles to reach the searched object (the green bottle).

The third environment is a real office room with a size of 650 cm x 650 cm. In Figure 4 one can see a snapshot of the room. It has many complex properties, which are too difficult for standard computer vision applications. It is unstructured, has different light sources and varying illuminations, unexpected objects, moving obstacles, etc. This is the most challenging setup for a robot's searching task.

Figure 4 The real office environment. The environment is very complex. It has many different objects and is non uniform illuminated. The sunshine through windows produces bright light stripes and the desks produce shadows.

1.3 Camera Image Transformation

The Robertino robot is provided with an omnidirectional camera (Figure 5). Due to the 360° view, the image is distorted. To control the robot by a human only with a visual input as sensor, it is preferable to obtain an image without distortion. Figure 6 illustrates the distortion removing process. Unlike other methods – like Tsai (1986) –, IDSIA uses a very simple distortion removing algorithm, because the transformed image will not be used by sensitive computer vision processes, which require an accurate input.

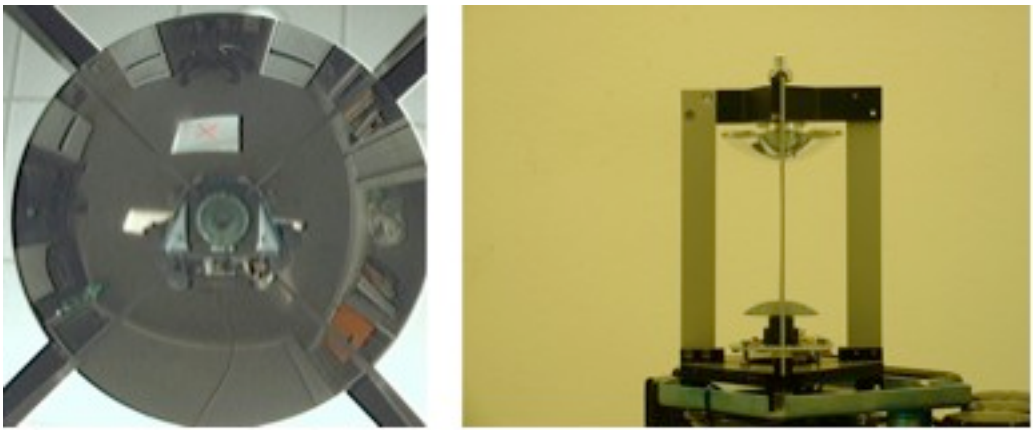


Figure 5 The omnidirectional camera. A FireWire web cam looks upwards into a spherical mirror (right) and receives a distorted image of the 360˚ environment (left).

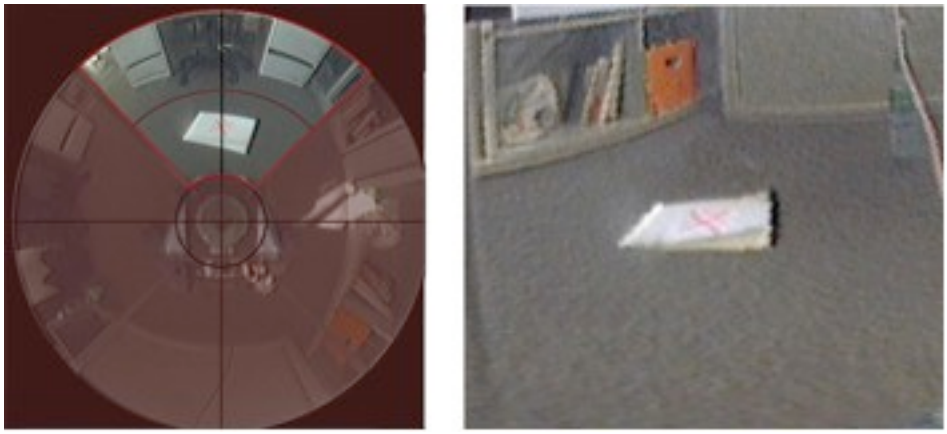


Figure 6 The distortion removing process. A 90˚ part of the distorted image is selected (left) and transformed to a rectangular bitmap (right).

The non distorted image is used to build the data set of the fovea: the image is transformed into several parts of different resolutions. The center of the fovea has the original resolution. In the outer parts the image is subsampled in several steps. Figure 7 shows a picture of the result of the fovea transformation. The input data is reduced from 36608 pixels to 429 pixels. If the fovea is not centered in the image, the outlying pixels are black.

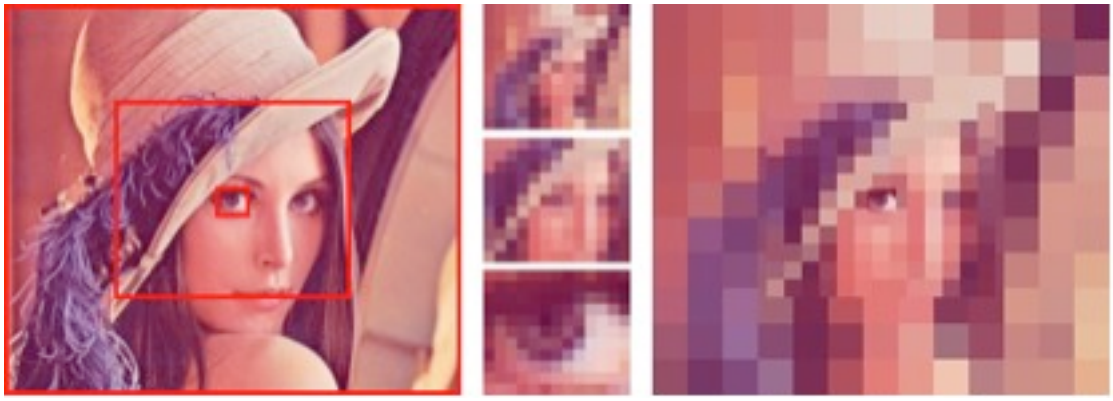


Figure 7 Fovea transformation of a photograph (right). In the center of the fovea is the highest resolution; at the border is the lowest resolution. The original image is divided into three regions with different resolutions (left). Every region has been sub sampled to a size of 13 x 11 pixels (middle).

1.4 Software Framework

The Robertino robot runs under a standard Debian/GNU Linux operating system. Some drivers for special hardware were added to the system: one for the CAN bus interface on the robot and one for the FireWire (IEEE 1394) interface. Especially the CAN bus driver is a non standard PC component. The web cam is connected with the PC over the IEEE 1394 interface. The system uses the Video4Linux library to grab pictures from the camera. The CAN bus is used to communicate with the motor driver, the tick counters, and the infrared (IR) distance sensors on the robot.

The framework for programming the robot consists of two separate C++ based application-programming interfaces (APIs). The first is hardware-close and is designed for developing fast applications with a low system delay on the robot. The second API is intended for development on an external workstation. This interface uses a client/server architecture to connect the robot with a workstation. The client API is available for Linux and Windows. After the completion of the development phase, the software can be ported to the robot to achieve a fully autonomous robotic system. The chart in **Errore. L'origine riferimento non è stata trovata.** gives an overview of Robertino's software system.

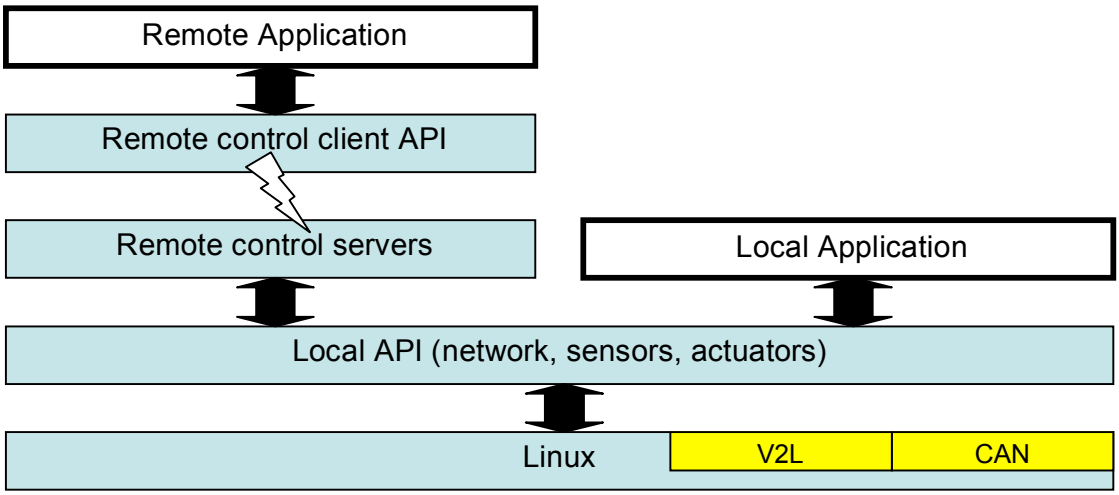


Figure 8 Robertino's software components. Two programming frameworks are available: one local API and one network client API.

The Robertino software is not well documented yet. The web site for on-line documentation is often off-line or improperly configured. Only one technical report from Verbeek et al. (2004), the header files for the APIs as well as a few programming examples describe the software system. A small overview of the APIs is provided in the next paragraphs.

1.4.1 Local API

The local API comprises three modules: the motor controller and sensor unit (*moc*), the vision unit (*vision*), and an operation system independent communication unit (*com*), which can be used to communicate with other robots or external computers.

The communication unit contains two classes: a *Socket* class to establish a connection and to read and write raw data of any length and a *ComData* class to convert pairs of names and values to raw data and vice versa.

The vision unit hides the complex initialization of the camera setup and provides a method *Vision::capture2* to grab an image. Furthermore, some methods for reading camera parameters and calibration are provided.

The *moc* unit contains a class *MotorControl* with methods for reading infrared sensor information, setting velocity commands for the robot, and reading collected odometry information from the robot. The calculation from robot velocities to motor currents is conducted by the API. Similarly, the Euclidean odometry information is calculated from the motor tick counters by the API.

1.4.2 Remote API

The server side of the remote API is based on the local API. Three server daemons run on the robot: *drived* (for sensor and actuator communication), *visiond* (for camera services), and *ctrld* (for shutting down the robot and restart services). The corresponding classes in the client API are *DrivedCom*, *VisiondCom*, and *CtrldCom*. The classes are based on Qt from TrollTech (www.trolltech.com). Qt is a C++ programming library, which is independent from the operation system and contains classes for communication, graphical user interfaces, file handling, and much more. It is available for Windows, Linux, Mac OS, and some embedded systems. It uses a so called signal/slot mechanism to send messages through the system. The client classes of the remote API use only Qt's signal/slot mechanism and socket API.

1.4.3 Server Modifications

The bottleneck for the remote control of the robot is the image transfer from the robot to the external computer. Without compression, transfer rate is limited to 4 images per second. Therefore, the API provides a jpeg compression parameter. However, jpeg compresses the whole image, but we want to use a fovea with a maximum image quality in the center of it and lower quality towards the borders. The fovea itself reduces the needed bandwidth significantly. Therefore, the distortion removing process must be executed on the robot.

1.4.4 Client Software

The client software is derived from the Robertino program *robomon* (robot monitor). The software is modified to handle fovea specific commands for the movement of the fovea. Moreover, an interface to control the robot with a joystick and to control the robot with AI software is added to the system. Figure 9 shows a screen shot of the software.

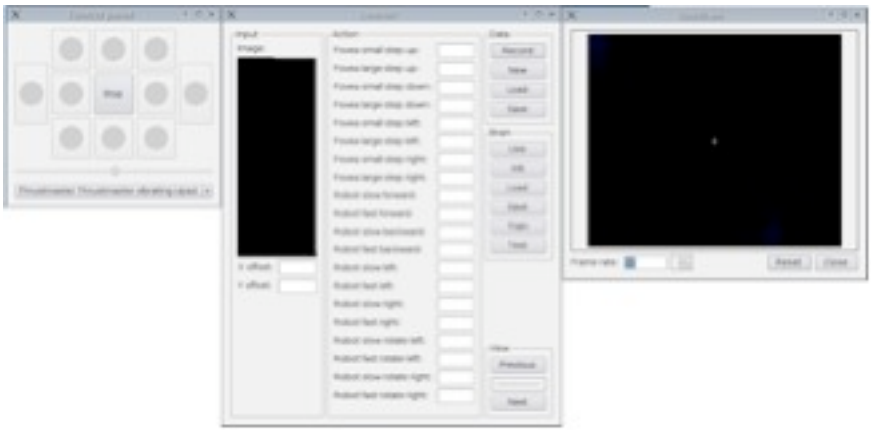


Figure 9 Screenshot of the client software. The manual control of the robot is on the left; the fovea image is shown on the right; the learning control interface is in the middle.

1.5 Simulated Vision

To accelerate the development of learning algorithms and to boost the learning time IDSIA will also work with some levels of simulation with different complexity. To start with an abstract one-dimensional simulation with a moving fovea in a one-dimensional environment.

1.5.1 1D-simulation

In the simplest simulation, the fovea deals with a one-dimensional environment. The robot stands still and can move the fovea left or right to focus on different objects to find the desired one. IDSIA has implemented a simulation of a one-dimensional world. The world consists of 27 bins in a line. Imagine a bin as a single pixel or a conglomerate of pixels. Every bin is able to contain one object. The fovea consists of 7 sensors in a line. Two of them – the outmost ones – consist of 9 bins and other two of 3 bins. The three sensors in the middle are mapped to one bin each. Only the centered sensor can distinguish between different objects. All other sensors can only recognize if there is at least one object in the respectively mapped bins. Figure 10 shows an example of a 1D environment.

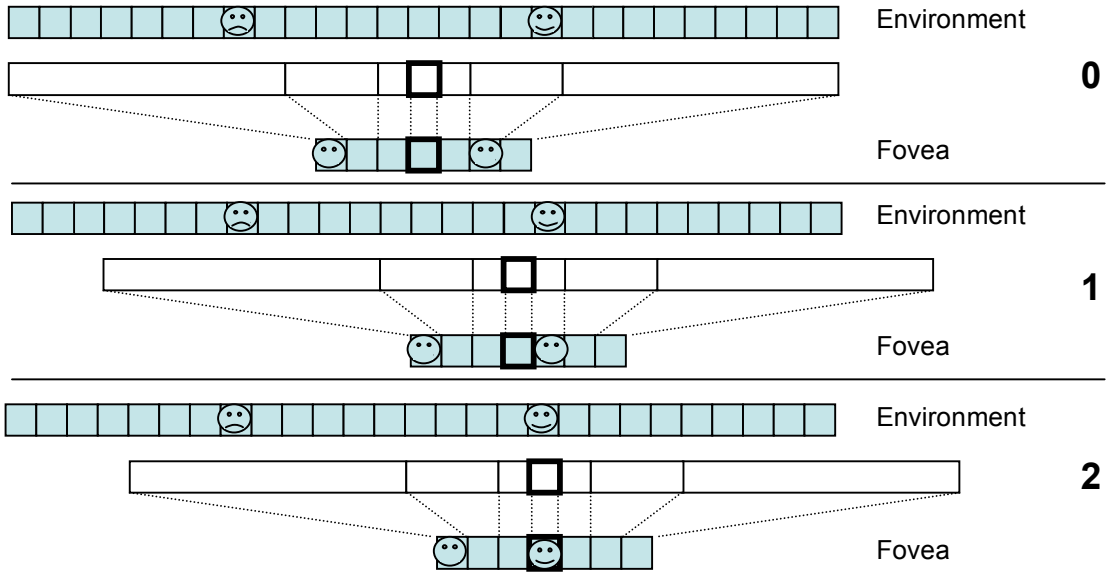


Figure 10 1D fovea in a 1D world. The detector of a specific object is only in the center of the fovea. Therefore the robot must focus on an object to recognize the object type.

The interface to interact with the simulation is simple. There are 5 procedures to initialize, provide actions, and receive fovea sensor data. Table 2 describes the interface functions.

void init ()	Initialize the simulation. This function must be called ones at start-up.
const vector<double>& getObservation ()	Returns the fovea input and some other agent dependent inputs, e.g. position of the robot.
void useAction (vector<double> action)	Calculates the state of the world at the next time-step with respect to the current action of the agent.
void reset ()	Generates a new world randomly and reset the agent.
bool isFinished ()	Is true, if the agent has reached the target.

Table 2 Fovea world interface. Five procedures are available to control the simulation

1.5.2 2D-simulations

IDSIA implemented two kinds of 2D-environments. The first world is an extension of the 1D-simulation. The image size is 27x27 bins and the fovea consists of 25 sensors. Figure 11 shows the relationship between the input and the fovea. The interface is the same as in the 1D case except for the size of the input and output vectors.

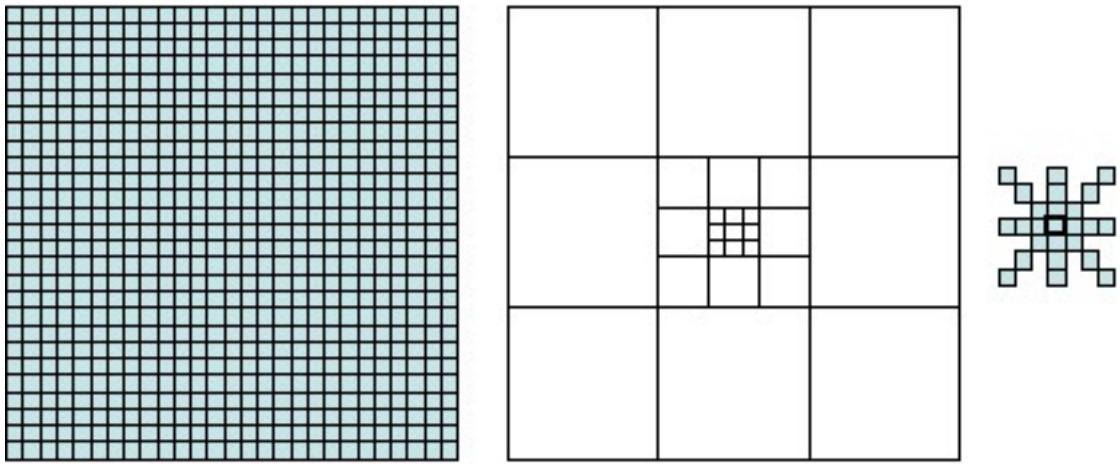


Figure 11 2D fovea in an abstract 2D world. Like the 1D-world, the detector of a specific object is only in the center of the fovea. The 729 input values from the image are reduced to 25 values of the fovea.

The objects in the two previous simulations are abstract in terms of their properties. The objects have only the attribute they are either the ones we are looking for or not. However, the second 2D-simulation uses an image with visual objects. The objects are triangles with one angle up or down (see Figure 12).

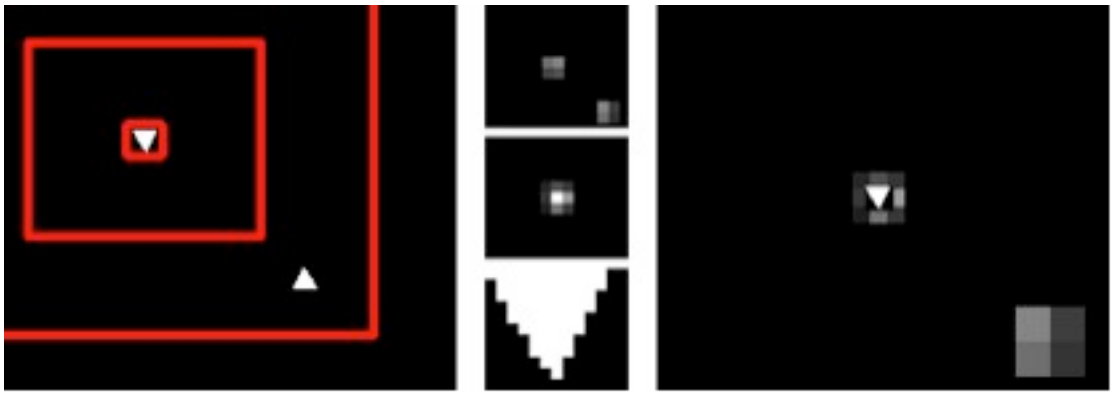


Figure 12 2D fovea in a visual 2D world. The fovea has it focus on the left triangle of the image (left). In the lowest resolution (upper image in the middle row), the objects have the same shape. Only in the area with higher resolution (lowest image in the middle row), the orientation of a triangle is visible. The right image shows the composed fovea image.

1.5.3 3D-simulations

With respect to the real world environments, IDSIA also uses two different 3D-world environments: one with simple objects (colored boxes, cylindrical shapes, balls) and one with objects from an

office environment. Like in a real world environment, the (simulated) robot has to deal with issues such as occlusions and view-distorting shadows. Figure 13 shows an image of a possible simple box-world. Therefore, both the simple environment and the office environment will eventually be designed such that they match better the real-world environments.

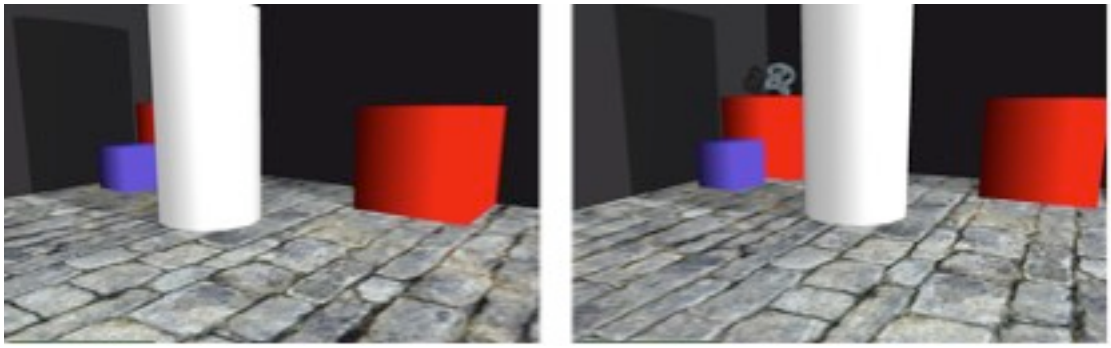


Figure 13 3D simulation of the box-world, produced by Ogre3D. This figure shows a room with some colored boxes and a cylindrical obstacle. In the left figure, the target object (the knot on top of the red box) is occluded by the cylindrical obstacle, so the robot has to deal with partial observability. It has to remember what it has seen before and where it has been in order to make a good decision on where to look and go next. Moreover, it has to rely on expectations in order to find the target faster (e.g. it could learn that targets tend to be on top of red boxes).

A simplified physics system is adopted, because more realistic tools like ODE are, for now, too complex and too slow. Since the stress in this task is on perception (anticipation of information gain) rather than action/control, we do not need the physics to be very realistic. The only important feature is collision detection in order to prevent the robot from moving into other objects.

The 3D visualization of the simulated Robertino, on the other hand, needs to be realistic. The vision is based on Ogre3D (Object-Oriented Graphics Rendering Engine, www.ogre3d.org) which enables fast rendering of 3D environments in a user-friendly, object-oriented manner. The library is a real-time 3D rendering engine, is cross-platform (Linux, Windows and Mac), and works with both OpenGL and DirectX. The API of Ogre3D allows easy manipulation of all relevant features such as shadows, lighting, movement, camera settings, and image production in various resolutions and points of view.

The simulated robot has the same abstract control interface as the real robot. The commands are abstract velocity values for the directional movement and the rotational speed. Furthermore, it has the same fovea-based visual system.

IDSIA uses Ogre3D system to produce views at three different levels of detail (low, medium, high), corresponding to the three regions of the fovea, and use the resulting data as input for the fovea learning algorithms.

1.6 Conclusion

IDSIA has done all preparatory work to start with learning experiments at different levels of complexity with the common goal: finding a target object in a partially observable environment by producing a sequence of saccades or other movements until the target is centered in the visual field of a sensor like the fovea centralis. In the simplest environment, the fovea is a sensor in a one-dimensional array. In the most complex environment, the robot looks for the object in a real-world domain. However, the agents in all environments have the same kind of sensor, which reduces the huge input data to a manageable size: the fovea.

Some modifications to the robot and the environment could make sense. To bring the simulation of the fovea close to a real fovea, the camera can be attached to a motorized camera mount. A second camera with a zoom object lens can be added to the robot to imitate the high resolution area of the fovea. If it were necessary to use a more physical simulation of the robot, it is possible do use ODE with the OgreODE plug-in for visualization. Also, the real and simulated environment can be changed, if a task is too complex or too simple in the current environment.

2 IST

FINDING AND LOOKING FOR

- ***Fetch that object!***

This is a human-robot interaction task focussed on believability. In the room there are several crates lie scattered around, acting as obstacles between Aibo and its searched target.
The human throws a red ball into the next room, then turns to an Aibo robot and says: "Fetch!" The robot should run into the room and designs a plan to find the red ball. While searching the space, its attention is drawn to a small handkerchief whose colour is just as the ball it is searching for. With its ear pointing forward, Aibo starts running, waving its tail and barking in anticipation. However, as soon as the robot realizes it is a mere handkerchief, its ears drop back and its tail falls between its legs. With a disappointed face, Aibo starts moving back, its gaze wandering across the room...

This scenario tries to integrate emotion and anticipation to achieve the believability of an embodied agent. This scenario may also be extended with the presence of another robotic agent that "observes" the scene and given its relation to the Aibo, reacts emotionally.

So far, in order to achieve this scenario, IST has developed a set of tools that help the exploration of the believability of the agent in the scenario. These tools will here be briefly described, which include a robot, an editor of motions for this robot and a simulator of the domotic environment that the robot will be in.

2.1 AIBO

As the scenario focuses on the believability of the agent and believability is based on emotional responses from the agent, there's a need for a robot that can have an affective interaction with the human.

Figure 14 Two AIBO robots.

The robot that matches these requirements is AIBO from Sony. AIBO is a home entertainment robot with the form of a dog and the main reasons for its choice are the relatively low cost and the state-of-the-art hardware, which comes with large development support. No other robot available for research has its price/quality ratio nor its mind programming easiness and community support.

Another important reason is the affective potential of such a robot. It was developed with affective interaction in mind. AIBO's emotional expressiveness can be achieved through several ways: it can have body motions that express its emotions and it can show emotions through its facial expressions. This is achieved through a grid of LEDs of four colours, it can produce sounds, which can express an emotion. In addition, the off-the-shelf mind software can express AIBO's emotions through its behaviour at some level. However, as this is a black box it cannot be reused for our research.

The possibility of using the robot iCat from Philips to incorporate this scenario is being studied. iCat is a robot designed for human-robot interactions which can generate facial expressions and also has a camera and microphones. This way iCat has a lot of affective potential.

For the IST scenario, the objectives will be to use all of the AIBO's forms of emotional expression along with the anticipatory affective behaviour to reach the desired believability of the agent.

So far, a set of tools has been designed to help the development of the scenario. As emotion expression is an important part of the scenario and body expression one of the major ways to achieve it, there had to be a tool to develop complex motions. This tool is AIBO Editor, which is described next. Also when developing an agent on a real robot it is very useful to have a simulator for the robot and the environment so that real world implementation issues are abstracted and the agent behaviour becomes the focus of attention. So, to aid in the process of developing the scenario, a simulator was developed for the domotic house scenario with a virtual AIBO in it.

2.2 Motion and Pose Editor for AIBO

AIBO Editor is an application developed to create and edit AIBO's body expression through poses and motions.

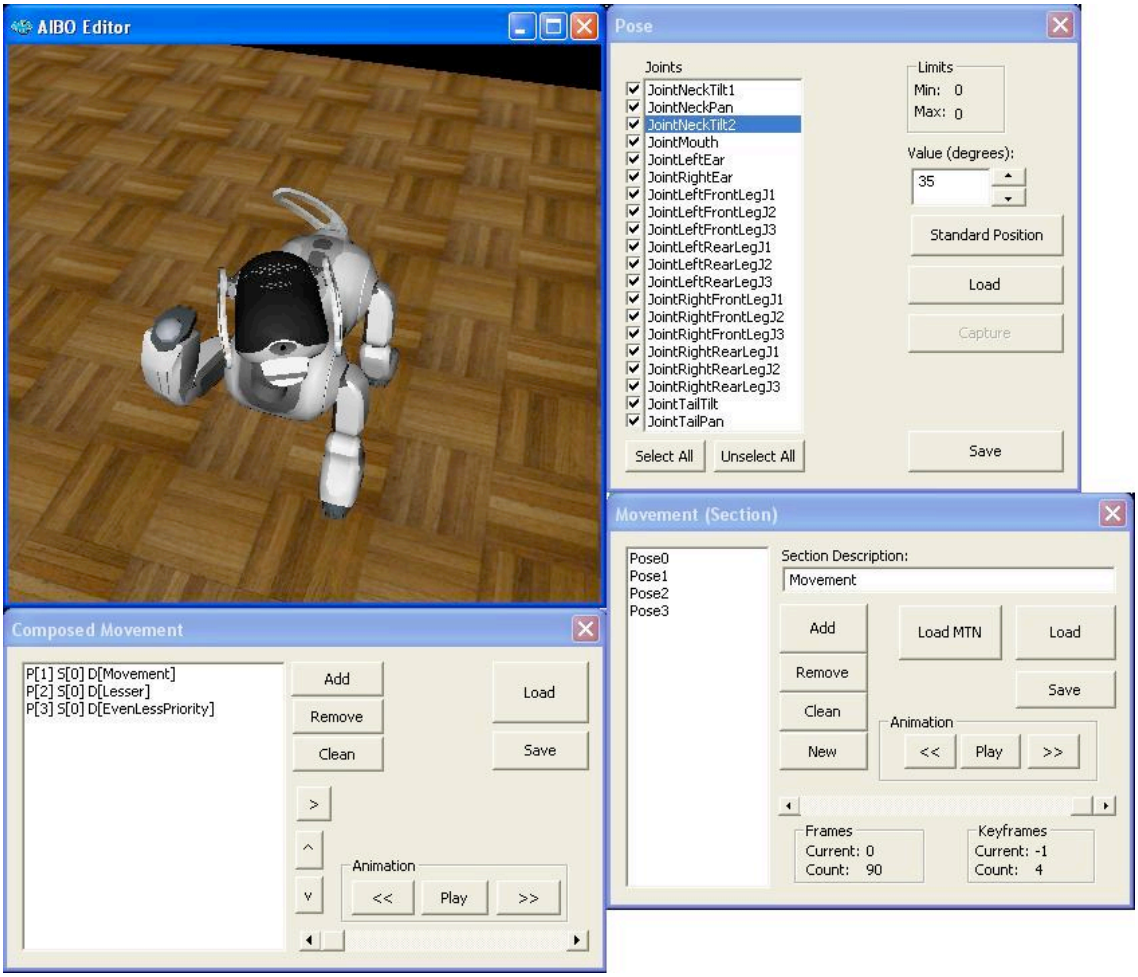


Figure 15 A view of the AIBO Editor tool.

Some tools, such as Skitter and Sony's Motion Editor, already exist for this purpose and are free of charge. Moreover, the tool has some differences that can be really important for the scenario implementation. So besides the usual poses and motions, like the other tools present, there are also composed motions, which are a composition of other motions reusing some parts of each one.

At the bottom of this hierarchy there's the AIBO *Pose*. A pose is defined by all the joint values of the robot. Above the poses are the *Simple Motions* like the ones created by other applications, which are just defined by an array of poses and generated by their interpolation along time. Then, there is the *Section Motion,* which is just a simple motion but it has the information on the high priority joints, that is, the most important joints for that motion. And above all there's the *Composed Motion*. The composed motion is a set of section motions and the order of priority between them in

getting resource locks for the joints. For instance, we could have a motion when AIBO is happily walking and one when AIBO is searching for something (moving its head around). If we want AIBO to happily walk while searching for the ball, then we can just say we want all joints except the ones related to the head from the walking motion and create a composed motion combining both (the walking and the searching motions). The walking motion is the high priority one.
Along with the editor there are a set of classes that allows the creation and manipulation of these special motions.

2.3 Environment and Robot Simulator

The domotic house simulator is an application called *Domo Simulator*. It simulates an automated house, the domotic control system of the house, its web server, and the robot in the house.

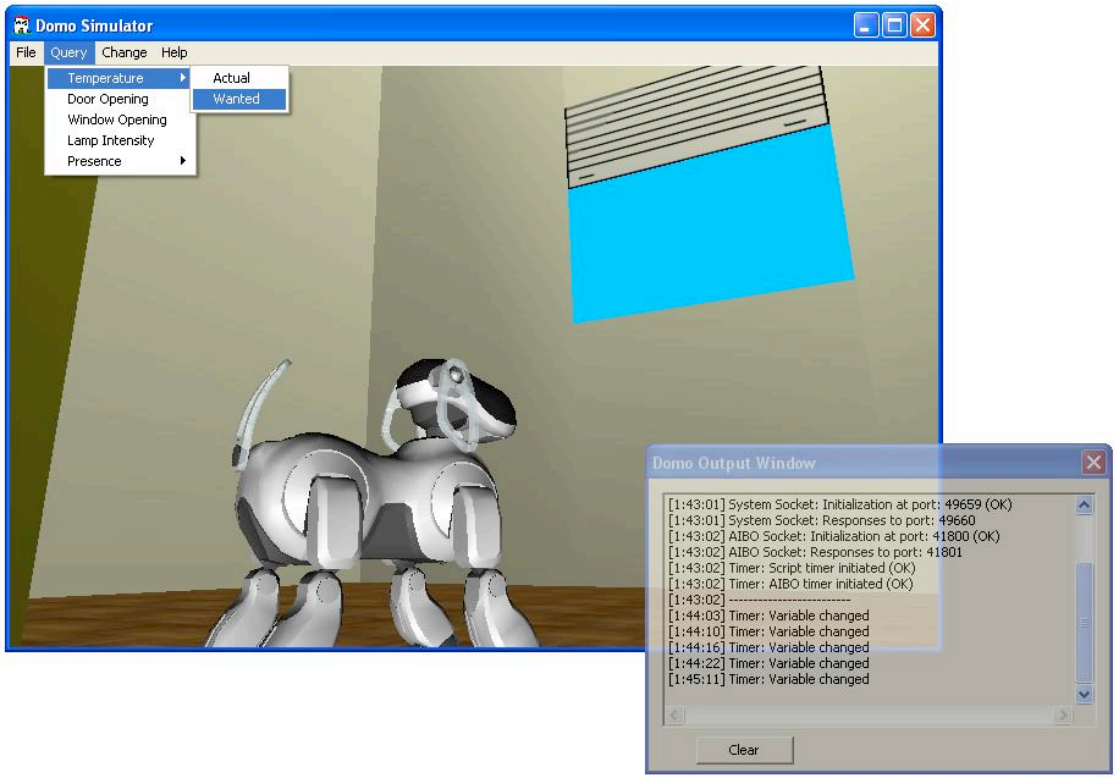


Figure 16 An overview of a virtual AIBO in the Domotic Simulator.

2.3.1 Environment Manipulation

The domotic environment is dynamic so the simulator tries to offer ways of changing the environment. These changes are made in variables of the environment like the temperature of a room, the desired temperature for that room, the lamp intensity, the door and window openings and the position of a person in the house. There are three options for domotic environment manipulation: a variable can be changed through the application menu, an action script can be loaded and executed and commands can be submitted through the web server to the domotic control system.

First we have to distinguish the kinds of changes in the values of the variables. A change can take place immediately or throughout time. Immediate changes should be avoided when simulating real environment behaviours, as a variable never changes immediately between two distant values (i.e. a door takes some time to close even if it's closed suddenly). This kind of change should only be used for debugging. Smooth variable changes are a linear interpolation between the actual value and the desired value at the given time. One can close the door in 3 seconds. As interpolation in the simulator is linear (at the time) a different one can be approximated to several linear ones.
Variable changes through the application menu are trivial. The variable to be changed is chosen. Then, the desired value, the interpolation duration and the subject of that change are indicated in the dialog box. The simulator, then, executes the change in the variable, which we call action.
Action scripts are text files with a list of actions and the time they should occur. Each action has the same variables as the ones specified in the menu (variable, subject, duration, value) plus the time they should start. Several actions can be executed at the same time so this is an advantage over the menu solution. Another advantage is that the environment behaviour can be reused and improved with little effort.
The third way environment can be changed is through communication with the web server. The web server of the domotic system has a socket interface based on the UDP/IP protocol so any application connected through a TCP/IP network can send messages to the server and therefore change the environment. This provides an improvement over the action script solution because the behaviour does not have to be hardwired from the start so it can be more dynamic. The protocol for messaging with the web server is described later.
These options for changing the environment are not mutually exclusive. All of them can be working at the same time. For example, an action script for fire in the kitchen can be running, an external application can be changing the user's position in the house (the user walking and extinguishing the fire) and the simulator user can go to the menu to change the temperature of a bedroom at the same time.
Also, the position of a person in the house can be changed through direct manipulation in the simulator through 3D navigation, which will be described shortly.

2.3.2 Visualisation

Since body motion is an important part of the scenario implementation, a robot simulator had to allow a realistic representation of such a robot so that people interacting with the environment could have an affective experience. So the environment and robot simulator has 3D visualisation and navigation capabilities.
One can navigate through the house just like in first person shooter games with a free camera or through the eyes of a user in the house or through the AIBO itself. As the camera is free, the house can be watched from any point like in the figure below.

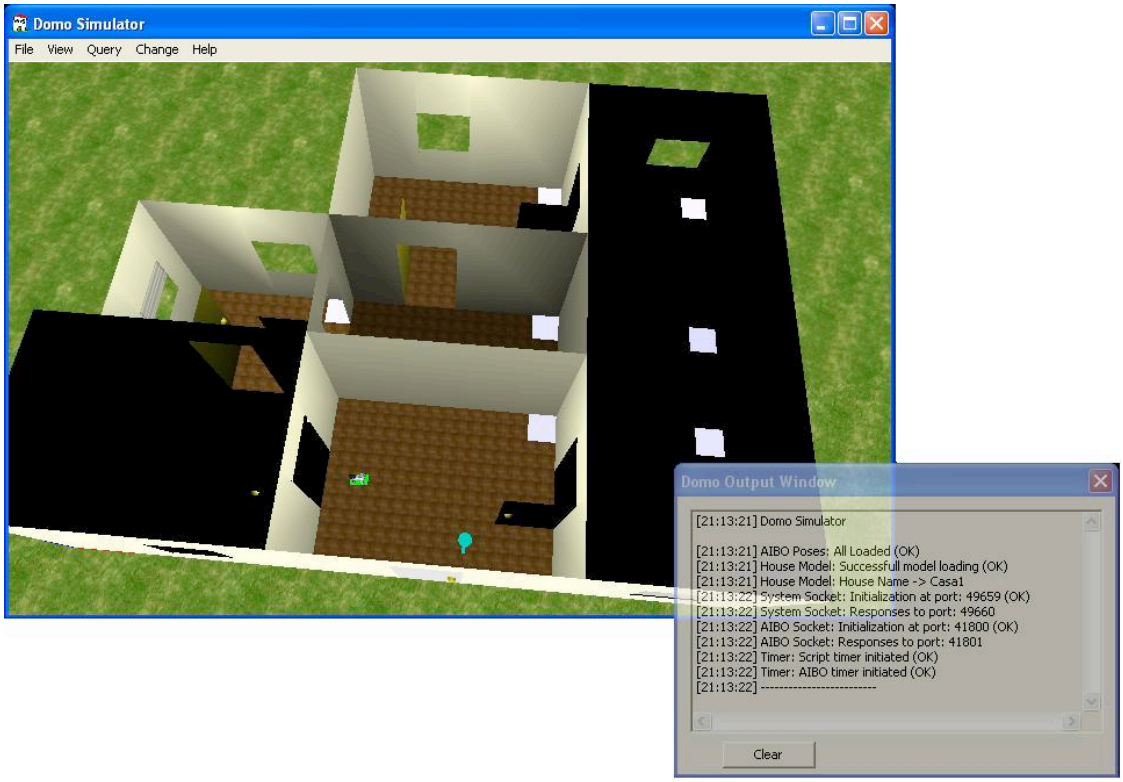


Figure 17 A view of the whole environment (house).

Also in the figure above hints of some variables values can be seen. An indication of the temperature of a room can be seen in the right topmost (north-east cardinal point) corner of the room. The temperature is indicated by the colour of the square: white for 22.5ºC, blue for 0ºC and red for 45ºC. There are shades of those colours between these values. Cyan is below 0ºC and black is above 45ºC. The lamp intensity of the room can be seen in the light of the room. Window and door openings can easily be identified. And the user and AIBO are in their positions, the user being the cyan lollipop.
These are just hints as they are not exact values. To see the exact value of a variable, the menu *Query* should be used or the web server can be asked to retrieve that value.

2.3.3 House Blueprints

The house blueprint is defined in a text file. This way, different room settings can be easily simulated just by changing house definitions in the blueprint file.
Houses are defined in a Cartesian coordinate system space where each unit represents one metre. The direction of the YY axis is towards the north and the direction of the XX axis is towards the east.
Each room has to be surrounded by exactly four walls and each wall must belong to a maximum of two rooms, each in a different side of the wall, meaning that there can not be diagonal walls and the rooms are arranged in a kind of grid where each row or line can have its own length.

2.3.4 Communication with the Domotic Web Server

As described earlier, the simulator application has an emulated web server. This web server provides an interface between the domotic control system (and the environment) and any other entity with access to the application through the UDP/IP protocol. This entity can be the AIBO robot, an application implementing AIBO's mind, a web page or an application controlling the dynamics of the environment.

Over the universal transport protocol is a messaging protocol for the domotic web server based on ASCII human readable messages. This protocol can change variable in the house just like an *action* described earlier, and it can be used to get information from the environment. The protocol messages are listed in the table below:

Message	Description
(SET <Subject/Device> <Property> <value>)	Change a variable in the house
(GET <Subject/Device> <Property>)	Get a variable value from the house
(FORWARD <Subject/Device> <Property>)	Request variable information whenever a variable changes
(UNFORWARD <Subject/Device> <Property>)	Cancel a previous continuous information request
(FORWARD-TIME <Subject/Device> <Property> <msPeriod>)	Request variable information every period time
(REPORT <Subject/Device> <Property> <value>)	Response from the server to an information request
(PROTOCOL)	Ask the server which protocol is using
(PROTOCOL-RESPONSE <ProtocolName> <CaseSensitivity[Y/N]>)	Server response to the above message

This protocol is meant to be aimed at a user of the house so changing the room temperature does not make sense but this option was kept so applications can control the whole environment.

2.3.5 Virtual AIBO Control

The AIBO in the house is controlled through a mechanism that is similar to the one in the web server messaging protocol. The application provides a socket interface based on the UDP/IP protocol so that any application can control the AIBO's behaviour.

The emphasis of this control is based on the visualisation of the AIBO's behaviour for affective evaluation so the control is not physical as in a real AIBO. The robot joints are not controlled individually, the motion of the robot is based on the motions defined in the AIBO Editor application and the AIBO's movements through space are not a consequence of these motions as it would happen in a real environment. The movement takes place by interpolating the AIBO's positions and orientation.

The high level protocol of control is very similar to the one for the house and its messages are described next.

Message	Description
(MOVE <X> <Y>)	Move the AIBO from actual position to X,Y at constant speed
(STOP)	Stop a movement
(GET-POSITION)	Ask the simulator for the position of the AIBO
(ROTATE <DegreesFromNorthDir>)	Rotate the AIBO to a given direction
(GET-ORIENTATION)	Ask for the AIBO's rotation
(SET-POSE id)	Set a specific pose for the AIBO
(SET-ANIMATION id)	Set an animation to for the AIBO
(PLAY-SOUND id)	Make the AIBO play a sound
(REPORT-POS <X> <Y>)	Answer from the server to a (GET-POSITION)
(REPORT-ROT <Orientation>)	Answer from the server to a (GET-ORIENTATION)
(DONE-MOVE)	Sent by the server when a movement takes place
(DONE-ROTATE)	Sent by the server when a rotation takes place

The mind of the AIBO is then implemented in an external application to the simulator and controls the virtual AIBO through these messages and the house variables through the web server.

2.4 Conclusion

So far, IST has developed a set of tools to be used in the implementation, testing and analysis of the AIBO agent. While the simulation provides fast and problem-focused testing and analysis, the real AIBO provides more affective experiences and rich environment accesses. The Simulator could be improved to approximate the simulation to real world interaction. However, the development of the real AIBO hasn't yet taken place.

3 ISTC-CNR

FINDING AND LOOKING FOR

- ***Finding a specific object (Game Room)***

The purpose of this task is to find and reach a specific object in the environment (e.g. a red cube). The degree of detail in the description must be sufficient to define unambiguously a single object, not a class of similar ones. For example "red cube" is to be used in the case when there is a single red cube, and "big red cube" if there are several red cubes with different sizes and only one of them is big.

- ***Finding members of a class of objects by class description (Game Room)***

The purpose of this task is to find any object matching some general or partial description (for example "find a cube" or "find a red object"). As in the previous case, prediction or anticipation can be based on previous experience, recurring spatial relations, etc.

- ***Looking for an object in a "dangerous" House (House)***

In this task the robot is looking for a target object in a House where there are also dangerous objects. The task is designed to explore specific relations between emotions and anticipation.

GUARDS AND THIEVES

- ***Conflict in accessing the valuables - simple (House)***

This task involves two agents – one thief and one guard. In the beginning several valuables are hidden in at least two different places or there are several accesses to the hidden place, in order to make the guard's task non-trivial. The session ends either when the thief has collected or found all the valuables or when the guard has arrested the thief either by blocking him or by touching him.

- ***Conflict in the access to valuables - complex (House)***

This is a social task involving several agents – several thieves and a guard. The session ends either when all the valuables have been collected or found (no matter by whom) or when the guard has arrested (caught) all the thieves as described in this scenario.

ISTC-CNR is involved in two scenarios (FINDING AND LOOKING For and GUARDS AND THIEVES) reflecting its consolidated research tradition in two different (both complementary and competing) approaches to cognitive systems: the bottom-up, sensorimotor approach, and the top-down, conceptual one.

On the side of the sensory-motor approach, ISTC-CNR will carry out research to build a controller capable of autonomously learning a repertoire of actions to be used as building blocks to produce more complex behaviours. The controller will allow a robotic arm (simulated and real) to: reach different target points in space with the tip of its last segment ("hand"), assume different postures in space, grasp objects of different shape, move objects in space. The system will accomplish these actions mainly on the basis of proprioception and information about position and shape of objects in space: the latter information will be given to the controller "from outside" or it will be collected by the system through a camera.

On the top-down, conceptual side, ISTC-CNR aims at a complete understanding of the deliberative or intentional control of action. In particular the research objective is to provide the cognitive system with (1) different control strategies based on anticipatory mechanisms at different levels of abstraction (ranging from deliberation and practical reasoning to routinary actions) (2) the capacity to rely on other cognitive systems on the basis of the prediction of their behavior and (3) the interplay between deliberation and anticipatory emotions.

3.1 The FINDING AND LOOKING FOR Scenario in detail

ISTC-CNR interpretation of this scenario involves a multi-joint robotic arm (both real and simulated) endowed with "proprioception" (sensors to detect current reciprocal position of joints in space, or a method to return similar information on the basis of vision) and, for some tasks, a camera. The goal of the research is to design and implement controllers capable of building a repertoire of actions to be used as building blocks to produce more complex behaviours. Each action of the repertoire is "organised" around a specific *anticipated desired state* that the system should be capable of achieving by means of the action itself. The research will also investigate the possibility of using *forward models* to enhance the process of learning the actions of the action repertoire. The research will start to tackle these issues on the basis of the working hypothesis that natural systems develop (during evolution and/or by interacting with the environment during life) a basic repertoire of actions that allow them to assume different *postures* in space with their limbs (this hypothesis is supported by empirical neuroscientific evidence in humans and other animals).

In order to investigate these issues, ISTC-CNR will work with simulated and real robotic arms. The reason is that manipulation tasks, contrary to navigation tasks, require the formation of a rich repertoire of actions in order to control the actuators so as to suitably interact with different, and possibly "rich", environment's states and objects.

Working with robotic arms is much more challenging than working with mobile robots. The reason is that manipulation requires robotic arms that tend to be mechanically more complicated than mobile robots. This implies two "costs":

1) The economical cost of robotic arms tend to be higher than the cost of mobile robots: given a level of precision/reliability of the systems considered, prices of robotic arms tend to be about 5 to 10 times higher than mobile robotic "bases".
2) Difficulties to have mechanically reliable robotic arms (in comparison to mobile robotic basis with a similar economical costs) given the resources available.

These problems have been solved in two ways:
1) ISTC-CNR started a research collaboration with the project RobotCub, funded by UE Commission (Unit E5 "Cognition") that started 1 year ago and will last 4 more years (http://www.robotcub.org/; the Coordinator of the project is Prof. Giulio Sandini, University of Genoa). RobotCub has the goal of building an "open source" humanoid robot to be used by the research community. Part of the budget of RobotCub will be invested to create an infrastructure to allow other research labs to carry out experiments on the robots produced by the project. ISTC-CNR will collaborate with the project's Coordinator (University of Genoa) to test successful algorithms on the prototypes of humanoid robots that will be designed and built during the project RobotCub (some prototypes are already available).
2) ISTC-CNR will use a "budget" arm (cost: 5000€) for day-to-day prototyping and pilot testing. The arm is produced by ActiveMedia Robotics (http://robots.activmedia.com), and has 5 degrees of freedom plus a gripper (see Figure 22 and the available information at the following website: http://www.activrobots.com/ACCESSORIES/Pioneerarm.html).

The scenario (environment and tasks accomplished in it) is explained more in detail in the next section, while the details of the simulated and real robotic arms and cameras will be illustrated in section 3.3.

3.1.1 The environment and the tasks

The goal of the controller/arm will be twofold:
1) Interacting with the environment to build a repertoire of action: the actions might be the capacity to reach different target points in space, to assume different postures, to grasp objects having different shapes, to move objects in space.
2) Using the actions as building blocks to build more complex behaviors (actions) in a hierarchical fashion, for example: reaching different targets on the basis of different visual percepts, assuming different postures in correspondence to different objects in space, performing sequential movements (e.g., first reaching a blue target and then a red target), grasping objects and moving them in space.

3.1.2 Dimensions through which the difficulty of the tasks will be manipulated

The difficulty of the tasks will be tuned on the basis of a number of "dimensions":

- *Static/dynamic targets*: in some tasks the targets and objects will be static during each test, while in some other more difficult tasks the targets will move dynamically during each test (e.g., tasks involving tracing a moving target).
- *Position of static target(s)*: in tasks using static targets and objects, the position of the targets and objects in the environment will be either fixed in all the tests, or variable in different tests and stable during the single test.

- *Shape of objects to grasp*: this might range from spheres (that do not require changing the actions depending on the orientation of the objects) to cubes or bars.
- *Number of components of tasks*: the task might be simple, for example reaching a single target, or more articulated, for example reaching two targets in sequence.

3.2 The GUARDS AND THIEVES Scenario in detail

Differently, the GUARDS AND THIEVES scenario has been chosen to represent kinds of problems that need higher levels of cognition to be solved.
The scenario is composed of two distinct tasks. The former aims to explore the relationships between higher and lower level of action control in order to provide the cognitive systems both with the capability of acting in a rational way and of being tuned to the dynamicity and uncertainty of real environments (see the task *Conflict in accessing the valuables - simple*). The latter is most focused on detailing the role of expectations in higher levels of cognition and in social interaction (see the task *Conflict in accessing the valuables - complex*).

To meet these different research issues, ISTC-CNR has selected two different simulation frameworks (see below for clarification). The framework needed to solve the former task is suited to model a wider range of control strategies and is used to study the interplay between higher and lower levels of cognition. On the contrary, to solve the latter task, an approach focusing only on the higher levels of cognition is considered as the most relevant.

3.2.1 The first task

The agent is a simulated robot, whose actions are abstract driving commands. The environment is the House, as described in D2.1; it is composed of many rooms, corridors and doors (that can become open or close with predictable dynamics). The agent models the Guard, while it is assumed that the Thief is controlled by another agent not modelled in a complex way (it will be instead a simple routine-controlled agent, whose dynamics are predictable).

The doors and the Thief are the main sources of dynamicity of the environment; it is possible to tune the difficulty of the environment by manipulating the complexity of their behaviour, their predictability and their number. The main criterion of success for the Guard is to prevent the Thief (or Thieves) from stealing the valuables. Such items are kept into some locations (that are known to the Guard). There are also some other static obstacles such as cubes.

The "social" dynamics between Guard and Thief are not investigated here (nor it is assumed intentionality of the Thief e.g. for anticipating it by using theory of mind, etc.). This work is intended to be complementary to other partners' one, e.g. implementing the Thief, and offers an opportunity for the successive phases of the project, comparison and integration.

The scenario is implemented in the Gazebo/Player/Stage simulator (described later) both in 2D and in 3D. The choice of 2D or 3D, as well as the reliability of sensors and effectors offer the opportunity of tuning the difficulty of the tasks.

The House (see Figure 18) has a complex plan (involving many rooms, doors and windows, and a corridor) in order to offer interesting situations such as hiding places, positions where many rooms can be spotted, etc..

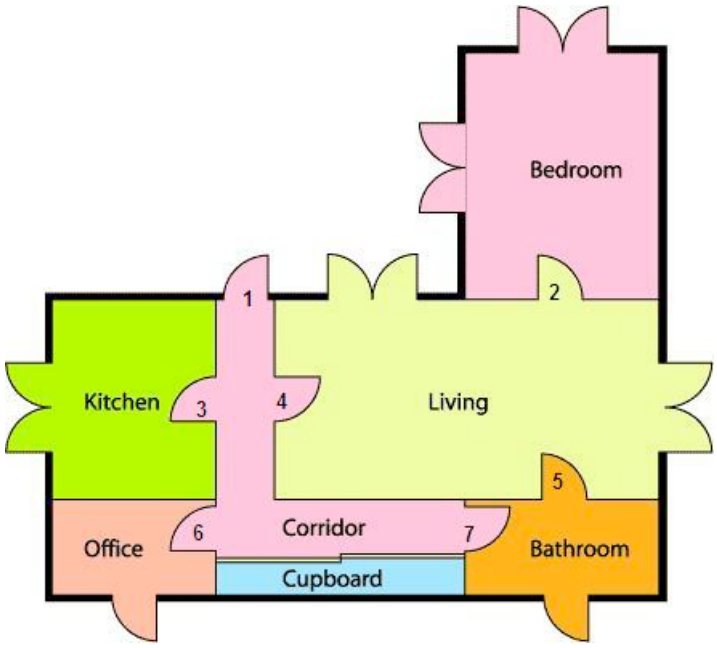


Figure 18. The House

The Architecture of the Guard

The architecture ISTC-CNR will implement is composed of many components, that are intended to realize a range of control strategies, including deliberation, means-ends reasoning, and sensorimotor interactions; moreover, the components interact with each other. The Guard's control strategy results, in fact, from an interplay between three distinct capabilities:

- *Intention Management*: This is the higher level decision process, inspired by the tradition in practical reasoning (Bratman 1987). This component selects among achievable goals with a process that takes into consideration their value as well as their satisfiability. The first main assumption is in fact that goals are selected on the basis of reasons, i.e. beliefs. Some of these beliefs are in fact explicit expectations, i.e. beliefs about future states (realizing the goals) depending on the agent actions (as well as on the dynamics of the environment, since some goals can be self-realizing). The second main assumption is that to an adopted goal corresponds now the activation of an Intention (intend to do a certain action/plan realizing the goal), having many additional roles with respect to goals and plans used for achieving them: intentions direct future processing, with a commitment on certain actions; intentions prevent inconsistent other intentions to be adopted; intentions provide a criterion of relevance about how to monitor the environment: the environment is in fact monitored with respect to intention success. Moreover, Intentions have specific dynamics, e.g. can be suspended, resumed, abandoned, etc.

- *Planning*: Once an Intention is adopted, the Planner selects/builds a suitable course of actions to achieve it, together with a monitoring strategy. There is in fact a functional continuum between intention adoption and planning, since an intention with no possible plans for the agent can not be adopted; moreover, an Intention already is about an action or plan. However, those action/plans are normally very abstracts and have to be partially or fully specified run-time. Moreover, separating the processes permits to model replanning as separated from Intention reconsideration: a new plan for the same intention can be run, if the previous one fails. Planning will be realized by the means-ends process. Plans will thus be produced in the form of a chain of actions (that will be realized at the lower level) and expectations (that will be continuously matched with perceptions).

- *Actuation and Adaptation*: this component is mainly responsible for the actuation of the plans by the means of sensorimotor interactions. The main components will be (fuzzy based) action Schemas (Roy 2005, Dresher 1991, Butz 2002, Wolpert and Kawato 1998) that are selected according to their expected consequences. In fact, schemas predicting better will be preferred.

The three capabilities are realized by three distinct components, that can however interact with each other. In many three-layer architectures, deliberate and reactive processes are executed in a separate way inside the different layers, resulting in different kinds of actuation. The components are thus not integrated but kept separated. On the contrary, one of the main architectural assumptions is that any action, even if deliberated and planned, in order to be realized has to be implemented by the means of low level, sensorimotor interactions.
The first kind of interaction between the components proceeds in a top-down way: after a phase of goal selection, an intention is passed to the planner; the planner selects a sequence of actions and expectations to be matched; those actions are realized and adapted by the means of sensorimotor schemas. However, there are many other possible interactions between the components, since in principle each process can introduce a pressure over the other ones; for example, an intention in action can be substituted by a suspended one (perhaps because some of its preconditions were false), providing that it is very urgent and its conditions are now met.

The roles of anticipation
In the different levels there are different (kinds of) expectations, having different format and roles. At the lower level expectations are directly matched with perceptions, retaining the sensorimotor format; moreover, they are often short ranged (describing the result of the next action). At the level of strategic planning, are also expectations about abstract properties of the environment that are not directly matched with perception, or at least not with a single observation. These expectations are often the main reasons to select among goals and plans, and are often long-ranged, describing the consequences of whole plans and even more. It is even very relevant to notice that at the deliberative level predictions are normally not matched with perceptions, but with goals, that are not current states of affairs and perhaps will never be. So, our work aims at modeling a range of possible roles of anticipation in a control architecture.

The Subtasks

The single agent architecture described here is intended to address two subtasks. It is assumed that the agent has an already available repertoire of actions (requiring predictive and anticipatory capabilities). For example, the task "*Recognition of the adversary among the moving (or moveable) objects"* is a precondition of all the other tasks into the scenario. It is assumed that the Guard is already able to recognize the Thief as well as other objects such as doors and rooms (implementing this capability by the means of simple routines). The complete repertoire of these capabilities (the *base actions*) will include features, objects and location recognition, as well as some other prior information (e.g. about distances or other relations between objects) that will be eventually introduced depending on the necessities. This approach offers also the opportunity of integrating the work of other partners addressing these specific capabilities, that is the next important issue of the Project.

- *Having two or more conflicting goals (e.g. protect two places), possibly conflicting, and arbitrating between them.*

The "goal arbitration" issue is mainly the work of the Intention Management component. Since the environment is dynamic (e.g. opening and closing doors; moving Thief), intention management has to depend not only on prior knowledge but on expectations. A case study will help illustrating this point (see Figure 18): the Guard is in the Living Room and it has two goals: *control the bathroom* and *control the bedroom*. If it expects that all the doors are open, it can choose for example to control first the bathroom. But if it knows that the doors 5 and 7 are closed (and will become open only after an amount of time that is sufficient to control room B), it can instead choose to control first room B.
Here the focus is on the peculiarities of intentions, including their roles as "drivers" of the system (for example avoiding contrasting intentions to be adopted); another very relevant issue is about their different dynamics: in fact, not all intentions are directly put in play (*in act*): some of them are *suspended*, waiting some conditions to be met; or put *in agenda*, expecting that some conditions will be met in the future; etc.

Arbitration is not limited to intention selection; even at the level of Planning there is the issue of choosing a plan among the suitable ones relying on explicit representations of actions/plans consequences. There is a problem of arbitration even at the lower level, that is very close to the behavior selection one. The arbitration is not based on explicitly represented reasons, but it is more associative. However, expectations play a very relevant role here, too: according to many schema-based approaches to control (Wolpert and Kawato 1998, Drescher 1991) the success of prediction of a Schema is the main reason to select it for action control, since it indicates that the Schema is well attuned to the current situation.

- *Integrating different levels of action control (e.g. routinary, reasoning), based on different kinds of expectations (e.g. implicit, explicit) and being able to arbitrating them by shifting level of control or by mediating.*

This task illustrates the need for a complex three-parts architecture, composed of different modules and capabilities. Our main questions are: *How are high-level "decisions" realized by low-level "behaviors"?* and: *How can the control come back from low to high level in case of necessity, e.g. errors?*

In real cases there is a continue interplay and shift of control between the levels of control. Consider for example that the Guard is in the living room and has two goals: *control the bathroom* and *control the bedroom*. It chooses to control first the bathroom (perhaps because it is closer and it expects all the doors are open). In order to realize its Intention, it plans a sequence of actions that include passing door 5. If door 5 is closed (assuming that the Guard did not expect this and can only verify this by trying to open it), this is a case of a failure at the level of actuation: it the Guard is not able to perform a given action that belongs to an intended plan, and a new plan has to be produced. This recover can be done by producing a new plan that includes passing doors 4 and 7. If door 4 is close, too, this is a different kind of failure, since it is now impossible to realize any kind of plan for the given Intention. Assuming that there is no subgoal possible (such as: *open one of the doors*), there is now the need for a new Intention (e.g. control the bedroom). Consider also that in this kind of situation some structures for maintaining knowledge are used in order to avoid the Guard to continuously move from door 4 to 5 and vice versa; consistently with our approach, only information that is relevant for the achievement of the goals is stored.

There are also possible conflicts between actions that are already in play and other goals, permitting the agent to *exploit opportunities*. Consider in the previous example another situation: if the Guard is (successfully) executing the plan to go to through doors 4 and 7 (the goal is to reach the bathroom), it can spot the Thief near door 6, in the office. At this point, it should not continue to pursue its original goal, but there should be Intention reconsideration (maybe leading to select a new goal, *capture (touch) the Thief*). This is an example of Intention reconsideration that is not due to a failure but to an opportunity.

Moreover, there is the possibility to plan actions that are impossible to realize for the agent (but of course this was not known during planning); a simple example is that the agent is too large to pass through a door. This situation again leads to re-planning, but for a different reason: not a failed precondition (a closed door) but a failed action, i.e. the expected consequences of one action are not realized.

A similar case of failed actions are actions that are not in the action repertoire of the agent: this offers the opportunity of asking other agents to perform parts of a plan, thus introducing social elements that is part of what will be addressed in the second task.

3.2.2 The second task

In order to deal with this task, the environment is populated with classes of unanimated objects (foods, fire, dangerous areas, walls, doors, path, tree, collection points, houses), and animated entities, e.g. the agents (Guards and Thieves).
The achievement goals between thieves are to obtain booties, both in an autonomous way and in organised coalitions. Differently, a guard has the goal to patrol areas and catch thieves. Thieves can

be caught when they moves close to the guard. The general layout of the environment layout is depicted in the following figure.

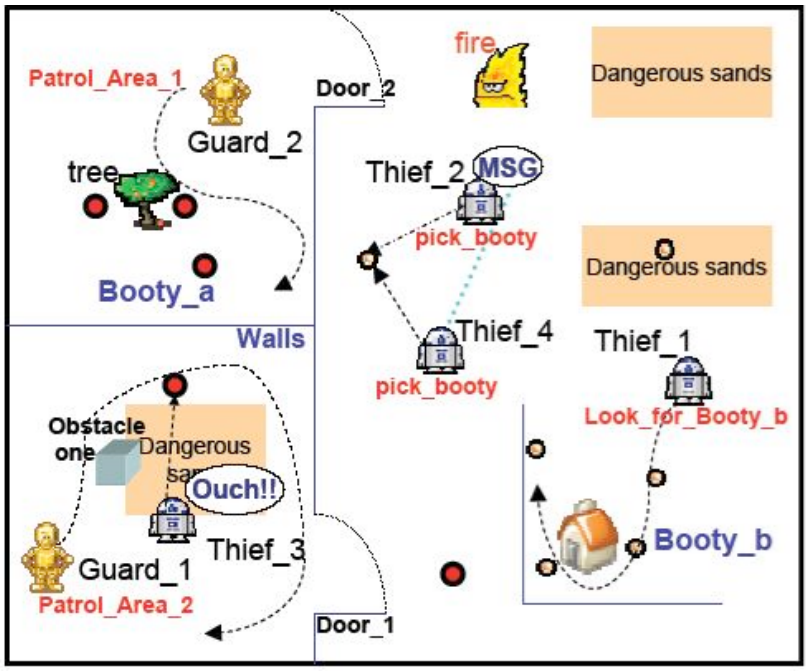


Figure 19 The environment layout of the second task.

The architecture

The core of the Belief-Desire-Intention architecture that is adopted in this task is described briefly in Section 3.4.2. Starting from the traditional approach to the BDI systems, ISTC-CNR is developing an architecture for dealing with expectations (and the emotional attitudes linked to them: relief, fear, surprise, etc.), learning and more complex cooperative and competitive attitudes (trust, reliance, delegation, help, etc.). At the top of the JADE-Jadex architecture, a new layer will be added, explicitly built to deal with tasks about agents with affective behaviours and social cooperation/competition.

Handling explicit expectations is a necessary capability for agents that have to predict both events in the world and the actions of their peers and that are to be endowed with emotional reactions to occurring events. Expectation lifecycle will be analysed from the origin (e.g. analyzing sources of expectation) to the final states (when expectation is tested, though perception, with the real world state, and possibly it is updated).

Expectations will enter the reasoning cycle not only as a simple value, but as “first class objects” modifying the traditional Reaction-Deliberation mechanism (which relies only on belief formulae i.e. logic expressions about beliefs) and including dynamic evaluation of expectations.

Since expectations have to be tested continually with the world, a component will be built to update expectations using the feedback signals filtered from the sensors. Including the expectations in the deliberation process, is a means to providing it with learning capabilities. In fact the more accurate the expectations are, the more the agents will be able to behave in the environment appropriately.

An emotion manager will be a very relevant component of the system. It is, on the model of the relationships between expectations and emotional attitudes reported in D5.1. A mismatch evaluator and an emotion manager will be adopted to deal with the last part of expectation lifecycle. On the basis of the (mis)matches between agent's expectations and the real perceived data, it is possible to enable both the triggered emotions (with their quantitative aspects) and the affective consequences on the agent's behaviour.

Another important component/function of the system is the plan recognizer. Assuming a shared knowledge about plans, agents can make plan recognition, in other words they can predict which action will be performed by an agent observed during action execution. Moreover, agents should be able to expect what the final, long term, state will be reached as the final goal state.

Finally abstract internal representations, through goal decomposition trees, are defined for the Action Observer components. Each goal will be stored in terms of triggered plans, and each of these plans in terms of sub-trees, where the leaves coincide with self contained actions. These representations, used for the schema-based recognition of other agent actions, contain only perceivable and observable data received from the world.

The subtasks
In what follows three subtasks are described highlighting the relevant issues.

- *Deliberating with expectations and coping with unexpected events*

In this subtask expectations will be matched-tested with the observable world state, through evaluation of data perceived from sensors. The agent will not only evaluate how accurate is the available prediction on the basis of feedback signals, but will also detect differences between what is expected and what is true in the real world. A level of surprise, for example, can be considered in terms of testing expectations in the world and quantifying the eventual mismatch. In this sense, surprise is a function of both (un)expectedness and mismatch – grade, directly appraised upon data's features.
Starting from some definitions of a set of emotions (surprise, fear, relief, disappointment and so on, see D5.1) this subtask is also intended to explore the consequences of the emotional attitudes.
An example about the use of expectations in the specific subtask is when a thief is expecting to find a booty in a specific place; on the basis of this expectation it decides (deliberates) to move itself towards that place of the world, then it is able to evaluate if its expectation matches with the perceived state of the world. On the basis of the matching result it should be able to feel different possible emotions like surprise (if the matching result is up a given subjective threshold). At the same way the thief could experience a relief emotion if it was expecting a guard following it (on the

basis of reasoning, indirect signals from the world, etc.) and it perceives from the real world the missing of the guard. And so on with other relationships between expectations and emotions.

ISTC-CNR will analyze both short and long term effects about emotions. As for the former kind of effects, we can say that, for example, surprised agents are characterized by a peculiar expressive behaviour (with social and communicative implications), they experience a momentary lapse in the control of execution and mobilize their computational resources. Sensors and any kind of available resources are immediately directed to the source of surprise (e.g. focusing on unexpected entities and objects) beginning short-term epistemic actions.
Analogously, for the long terms effects, again in the surprise example, there could be a general increase of the level of attention. Agents allocate more resources to epistemic actions and attentive processes, with direct effects in reducing promptness and speediness and side effects in bodily reactions, as energy consumption. Surprise has a direct effect on the level of agents' *cautiousness*. This can reflect on attentive capabilities (e.g. cautious agent engages in belief revision and increases at the same time control activities), and self-trust (e.g. prudence about expectations and uncertain beliefs, planning and intention revision).

- *Plan-based prediction of the others' behaviour*

More social in nature, this subtask requires mechanisms and capabilities to recognize, through observation or explicit communication, the intentions of other agents.

As in the classical approach to planning and plan recognition, the notion of action as step of a plan will be introduced to consider plans as procedural, activity-based processes, made by sequences of actions. Plans are viewed as flow charts where actions are the coarse grained nodes.

Agents will be provided with a structured knowledge about actions and plans and other contextual knowledge (agent's roles, world rules, etc.) to identify actions and infer the associated plans. Agents will be able to anticipate, with a reasonable gap of ignorance, a small set of action's effects. (e.g. seeing a Thief moving towards a booty, the Guard recognizes the plan "pick_up_booty" that has as main goal the result "to have booty").
In general, during action execution modifications will be made in the world: Generally a subset of these modifications can be captured in a window of observability where agents can recognize and attribute actions to specific plans (e.g. seeing a Thief moving and bringing an object allows other agents to deduce that the Thief is carrying the booty to the haunt).

Sometimes recognized actions can be included in more than a plan (there is an ambiguity for the plan recognition): in these cases recognizing agents has different possibilities:

1) waiting for the next action, hoping that it will disambiguate the plan to recognize;
2) increasing its own knowledge useful for the recognizing plan; this could mean to increase the attentive processes about specific parts of the world. In this case, the agent has to increase the computational resources allocated to find elements allowing action recognition and evaluating them in the context where the actions are performed (e.g. perceiving an agent with a key on a path at the end of which there is a door could be used to infer that the agent

has to pursue the next action close_door – if the door is open - or open_door – if the door is closed). Agents try to match such frame with the sensor data about other agents (e.g. if they are carrying object) and the environment (e.g. if there is an obstacle near to them). A "schema based frame" is built to evaluate contextual conditions against world knowledge.

3) recognizing agents evaluating some further contextual conditions like roles and reputation (e.g. a Guard near a Tree is in patrol_tree action, but a Thief near a Tree is looking_for_booty;).
4) recognizing agents using explicit communication: in favourable contexts message exchange can be involved in others behaviour prediction. Agents can perform a direct solicit-response messaging in order to directly know intentions (e.g. gangs of Thieves can natively communicate and share not only knowledge but also intentions).

- *Reliance, help, obstacle, delegation and trust both for competition and cooperation by anticipating other's behaviour (e.g. by removing/putting obstacles or doing/hampering part of other's work, asking another one for help).*

Once agents are able to predict other agents actions, recognize intentions and foresee future world states, they can behave in terms of anticipated world state distinguishing between positive and negative social interference (Castelfranchi 1998): this enables agents in reading the world in terms of opportunity/chances or obstacles.
Agents are able to engage in *cooperation* (exploiting the result of other agents' actions to enhance individual goals or to obtain a common goal) or *competition* (hindering and blocking other agents actions to prevent other agents to reach the same goal).

Predicting other agents' plans and their consequences can trigger intention revision in the agent. Agents can activate a meta–level reasoning to build an expectation about other agents actions. As a consequence, an agent can do *reliance* on another agent's action for example deciding of engage in direct action because someone else will realise the goal it is pursuing. The agent forecasts result states about other agent intentions or delegate without explicit request.
Let us show some examples. First of all we introduce the notion of *reliance*: Agent-a is relying on agent-b if there is at least an action of agent-b that is useful for agent-a and it decides to use that action in its own plan while agent-b is performing that action (Falcone and Castelfranchi 1998).

Cooperative Reliance:

- Guards patrol different areas, coordinating themselves.
  - Guard_1 has to patrol zone_a and zone_b. Guard_2 has to patrol zone_b and zone_c. (zone_b can be patrolled by both guard).
  - Guard_1 is going to patrol zona_b; Guard_2 remains in Zone_c, even if he would deliberate (in absence of the other guard's action) to change zone, until Guard_1 is in Zone_b.

- A special case is given by reliance on world *object* or *events*:
  - the Thief going to pick the booth makes reliance on the sun lighting the path on which it has to go on;

- if there is an obstacle between the thief and the guard, the thief could make explicit reliance on the obstacle for going to pick up objects (reliance for competition).

Competition:

- Thief_1 is going to eat Food_1. Thief_1 see Thief_2 going to Food_1 too. Thief_1 anticipates that Thief_2 will arrive to Food before it without external modifications of the situation. Thief_1 can choose between the plans:
  - To abort the Goal, Thief_1 drops the Goal when considers it impossible to reach (this decision permits to Thief_1 to save energy);
  - To choose a shortest path in order to anticipate Thief2 with respect to the Food_1;
  - To create an obstacle to Thief_2 (e.g. close a door and invalidate preconditions for action).

Cases of Help:

- Dropping obstacles for the others: e.g. Help Thief2 to realize precondition for the next action);
  - Suppose the case of *thief1* having the plan of collecting *tool1* and *tool2* before going to the *house* of the *booth*. While *thief1* is going to take *tool1*, *thief2* could take *tool2* and make it available for *thief1*.

3.3 FINDING AND LOOKING FOR Scenario

As mentioned in section 3.1, the FINDING AND LOOKING FOR scenario will use simulated 2D and 3D robotic arms and cameras, and real 3D robotic arms.

3.3.1 The simulated robot

Simulated 2D robotic arm. The 2D simulated robotic arm moves on a horizontal plane (Figure 20 shows a snapshot of the simulator). The arm can be composed of two or more segments and the same number of degrees of freedom. The simulator allows simulating only kinematics of arms without dynamics, or arms with more realistic dynamics. The controller issues commands to the arms in one of the following ways (the effects of these choices might be an issue of research): a) torques exerted by the motors; b) desired variation of angles between the arm's segments; c) desired angles of the arm's segments.

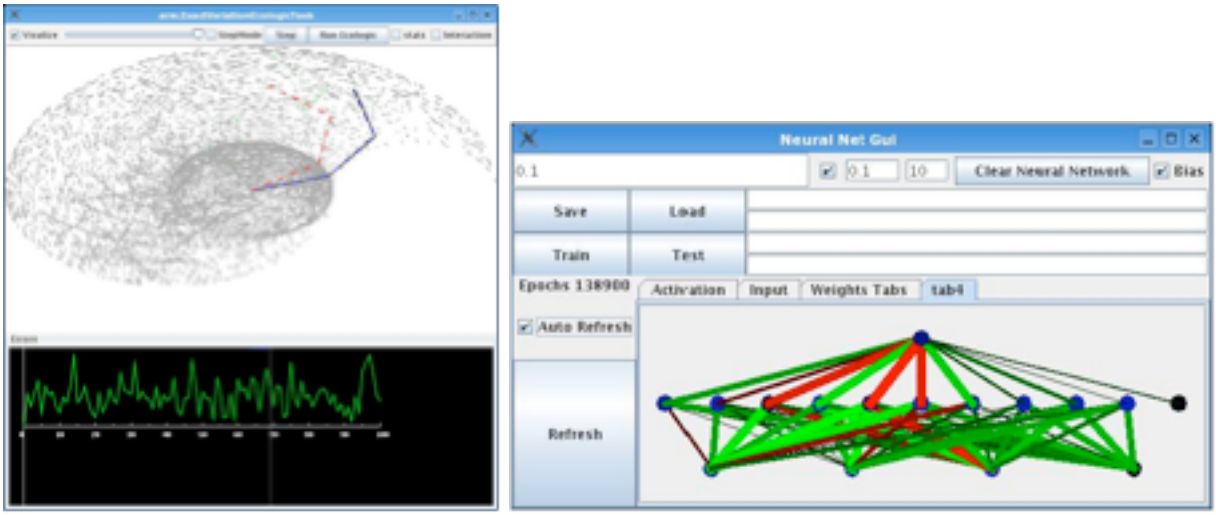


Figure 20 The simulator of the 2D arm. Left: a 2D arm with 3 segments while engaged in learning to achieve different targets in a reaching task (bottom left: error plot). Right: the interface of the simulator reporting a graphic representation of a neural-network controller.

Simulated 3D robotic arm. The 3D simulated robotic arm is composed of two segments, and possibly a gripper, with the following degrees of freedom: shoulder 3, elbow 1, wrist 2, gripper 2 (at least at the beginning, research will focus on tasks not requiring the gripper for grasping, see section 3.1). Commands are issued to the arm in ways similarly to what is done within the 2D simulator (torques, angles' variation, desired angles). The simulator of the 3D arm is based on "ODE - Open Dynamic Engine" (http://ode.org/), a free software licensed under the license "GNU GPL". ODE is a platform independent C++ library for simulating articulated rigid body dynamics, moving objects, ground vehicles, and robots with limbs. It supports advanced joints, contacts with friction, and built-in collision detection.

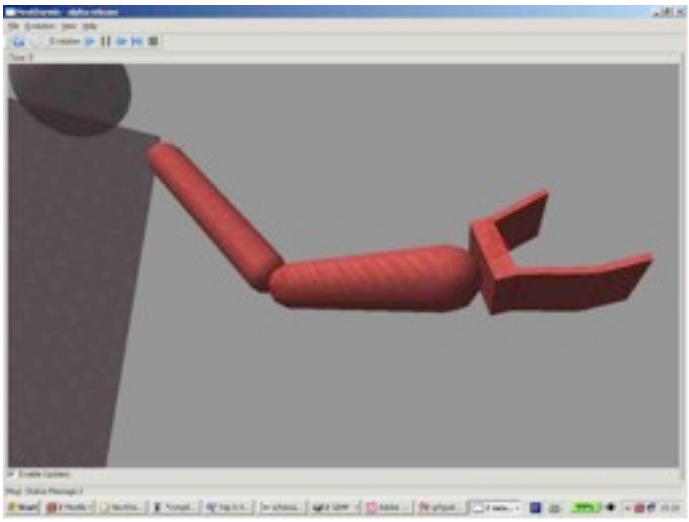


Figure 21: The simulator interface of the 3D robotic arm and the arm

Simulated camera. At the beginning ISTC-CNR will mainly work on the basis of proprioception and simplified simulated cameras, for example "retinas" that return "black and white" gross images (or more than one "overlapping" images if different colors are needed). Simplified simulated cameras (or "camera-like" abstract information) will suffice because research will mainly focus on organization of motor behaviour.

3.3.2 The real robot

The robotic arm used is a Pioneer arm^{TM}, produced by ActiveMedia Robotics (http://robots.activmedia.com), shown in Figure 22. Pioneer arm is a relatively low-cost arm for use in research and education. It has 5 degree-of-freedom and holds a gripper. The arm is driven by six, reversible 5v DC open-loop servo motors, and can reach up to 80 cm from the center of its rotating base to the tip of its closed fingers.

The arm can be controlled through a computer connected to it via a TCP socket. The whole system uses a client-server architecture: the robot hosts the server while the computer hosts the client applications. The robot's server (operating system) manages all the low-level details of the robot. Client applications can issue commands to the robot through API functions collected in a set of libraries ("ArAKIn"). These API functions can be directly used in custom C++ programs compiled and run either under Windows or Linux platforms.

Figure 22: The Pioneer 3 robotic arm^{TM} developed by MobileRobotics.

The experiments considered here will mimic proprioception sensors' readings on the basis of the commands issued to the arm (e.g., desired positions or angles of joints) or positions of the arm's joints in space inferred through an external camera.

Real cameras: ISTC-CNR plans to carry out integration work with other partners of MindRACES that are focusing on problems regarding vision and attention (e.g., IDISA and LUCS), in order to study eye-hand coordination problems. For this reason, it will adopt the hardware/software solutions developed by those partners.

3.4 GUARDS AND THIEVES: the simulation frameworks

As for the second scenario, ISTC-CNR has adopted the framework AKIRA for the first task and Jadex for the second one.

3.4.1 AKIRA

AKIRA permits a versatile management of the interactions among the components, by the means of both symbolic and connectionist dynamics. Differently from standard cognitive architectures such as SOAR (Rosenbloom et al. 1992) and ACT-R (Anderson and Lebiere 1988), where components and modules are cognitively impenetrable, in AKIRA it is possible to represent both symbolic and energetic exchanges between them (such as activation and inhibition, competition for limited resources, etc.). Moreover, according to our needs, all components and modules include both perceptual and motor processes; as stated earlier, they only differ for the degree of abstraction of the representation and control strategies they use.

AKIRA (http://www.akira-project.org/) is an open-source, C++ multithread framework that integrates features of Multi Agent Systems and Pandemonium (Jackson 1987); it has been developed by ISTC-CNR and NOZE and it has already been tested for many tasks (e.g. Pezzulo and Calvi 2005, Pezzulo and Calvi 2005b, Pezzulo and Calvi 2005c).
According to the Pandemonium metaphor, the kernel is called Pandemonium and the agents are called Daemons. Here we introduce briefly its main components.

The Pandemonium. The Pandemonium is the system kernel, the main process that instances the threads that are necessary to execute the Daemons (Agents) and that executes all the monitoring and control operations over the single components. Its parameters are configurable at start-up through an XML configuration file; it contains an XML description of: available memory; max number of executable threads; some features for Agents execution (e.g. priority, lifetime, resources); other system properties (garbage collecting, facilities for system and Agents debugging). The Pandemonium Cycle monitors the activity of all the Daemons (including exceptions) and is responsible for many system procedures, e.g. garbage collecting, showing the statistics for Agents, XML stream and system energy.

The Daemons. The Daemons are the atomic computational elements, each having its own thread and carrying its own code, that are initialized and executed by the Pandemonium during the system lifetime. Daemons are hybrid, having both a *symbolic* component (the carried operation) and a *connectionist* one (energetic attributes regulating the dynamics of the system). In fact, Daemons can share and spread energy throught an Energetic Network; moreover, a centralized pool of resources, the Energy Pool, gives an upper bound to the total amount of resources available for the computation, thus introducting a competition for limited resources among the Daemons. Figure 23 provides an intuitive picture of the concurrency model. The activation of each Daemon is calculated by an energetic network affording energetic exchanges such as spreading activation. Activation becomes priority of the Daemons' threads, driving their sequences of activation: more active Daemons can act more. Thus, Daemons are represented both as *nodes* (circles in the grid on the bottom), that exchange activation (via the links), and as *agents* (circles in the cloud on the top). The *priority* of the agents (their height in the cloud) depends on the activation of the correspondent

nodes. Daemons also share a limited amount of energy, that is represented by a centralized resurce called the Energy Pool.

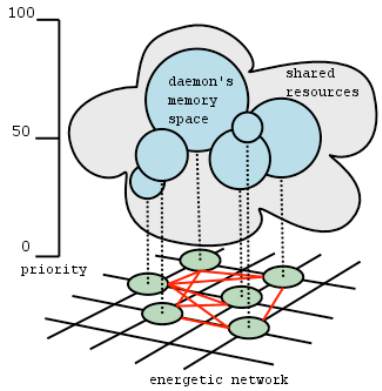


Figure 23. The two aspects of AKIRA

Writing Daemons in AKIRA. User-defined Agents inherit from an abstract Daemon declaration as well as from many pre-defined prototypes and models. Figure 24 shows the Agents generation process: the programmer extends some Daemon models; the Agents are dynamically managed by the Pandemonium and start their lifecycle as threads. The semantic imposed by the programmer to the Agents is specified in the *init()* and *execute()* functions. They are called by the framework as part of the run method and used as entry point for each Agent thread. Exiting from run means a regular termination of the current thread with the destruction of everything in its local space.

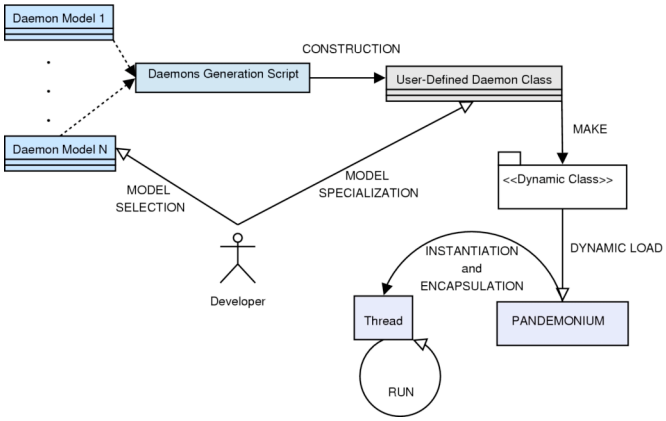


Figure 24. The Daemons generation process

Message Passing. In AKIRA three message passing mechanisms are available.

- The *Blackboard* (XML Stream) is a shared data structure divided into blocks containing AXL (AKIRA XML Language, a custom KQML-like data exchange language) packets, where the messages are concurrently written and read.
- *AkiraGenericObjectFactory* allows developers to create, set, get and destroy on the fly shared objects of any kind; it is the slower mechanism.

- *AkiraGlobalVariableFactory* is limited to shared variables of scalar type. All the mechanisms have a *templatized mutual exclusion* policy to guarantee thread safe and consistent access to all data.

Figure 25 illustrates the main components of AKIRA: the Daemons, sharing energy and spreading it via the energy network; the messaging infrastructure.

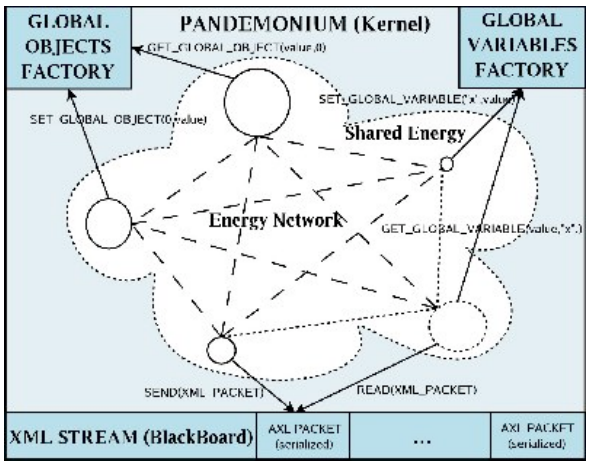


Figure 25. The components of AKIRA

In the design of the three-parts architecture of the Guard, many Daemons will be used. As a first approximation, each component will be implemented by using a single Daemon; they will communicate via the Blackboard and the control will be prioritized by setting different energetic values between them (e.g. Intention Management will have higher energetic resources). In successive steps beliefs, goals, plans and schemas will be implemented using many interacting Daemons, in order to permit more complex dynamics between them. For example, conflicting goals or schemas can be represented by using different Daemons, with different energetic values representing their urgency.

Requirements of the Guard Architecture

According to our architectural needs, the selected framework has to furnish three main features:

1. the possibility to design and implement different kinds of cognitive functions and capabilities, ranging from deliberative processes to routinary actions;
2. the possibility to integrate them and to model the dynamics of their interactions;
3. furnish the interface to a suitable simulator.

As for the first point, AKIRA includes a fuzzy logic library that has been used for implementing a schema mechanism (Pezzulo and Calvi 2005b) as well as a scripting language for treating domain knowledge (beliefs) and goals, including deliberation and means-ends analysis (Pezzulo and Calvi2005c). These facilities have been individuated as the main requirements for implementing the components of the Guard architecture.

As for the second point, AKIRA agents can be exploited for embedding the different kinds of processes (deliberate, means-ends, schemas). AKIRA Agents can communicate through a Blackboard and share energetic resources, influencing the priority of the related processes and making their representations more or less available to the other ones. The messaging infrastructure (that is XML based) will be used for example for representing the passage of a goal from the deliberative phase to actuation. The energetic dynamics of the Agents will be the basis for prioritization of control; for example, the different dynamics of slow-and-accurate processes (such as deliberation or strategic planning) or faster ones such as sensorimotor interactions.

Finally, AKIRA is interfaced with the simulator ISTC-CNR has selected for this task, that is described in the following paragraph.

The Environment and Physical Simulator
ISTC-CNR has also adopted the simulator Gazebo/Stage/Player (http://playerstage.sourceforge.net/), developed at the University of South California and used by a large number of researchers worldwide. The simulator has realistic physics (based on ODE) and includes three components:

- **Player**: Player is a device server that provides a powerful, flexible interface to a variety of sensors and actuators (e.g., robots). Because Player uses a TCP socket-based client/server model, robot control programs can be written in programming language and can execute on any computer with network connectivity to the robot
- **Stage**: Stage is a scaleable multiple robot simulator; it simulates a population of mobile robots moving in and sensing a two-dimensional bitmapped environment, controlled through Player. Various sensor models are provided, including sonar, scanning laser rangefinder, pan-tilt-zoom camera with color blob detection and odometry.
- **Gazebo**: Gazebo is a 3D, dynamic, multi-robot simulator. Whereas Stage is intended to simulate the behavior of very large populations of robots with moderate fidelity, Gazebo simulates the behavior of small populations of robots (less than 10) with high fidelity.

Software developed with Gazebo/Stage/Player can easily be ported into real robots, since many interfaces are furnished for many of them, including the Pioneer 3.

3.4.2 Jadex

JADE (Java Agent Development Framework; http://jade.tilab.com/index.html) is a Multi Agent Systems framework fully implemented in Java language. It simplifies the design of multi-agent systems through a middle-ware that complies with transport end-to-end interworking, interoperability and communication like FIPA protocols. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote interfaces. The configuration can be even changed at run-time by moving agents between agent containers located across networks.

Jadex (http://sourceforge.net/projects/jadex/) stands for "JADE extension" and represents a Belief Desire Intention (BDI) extension for the JADE multi-agent platform.

Jadex incorporates the traditional BDI model into JADE agents, by introducing Beliefs, Goals and Plans as first class objects, that can be created and manipulated inside the agent. In Jadex, agents have beliefs, which can be any kind of Java object and are stored in a belief base. Goals represent the concrete motivations (e.g. states to be achieved) that influence an agent's behaviour. To achieve its goals the agent executes plans, which are procedural recipes coded in Java.

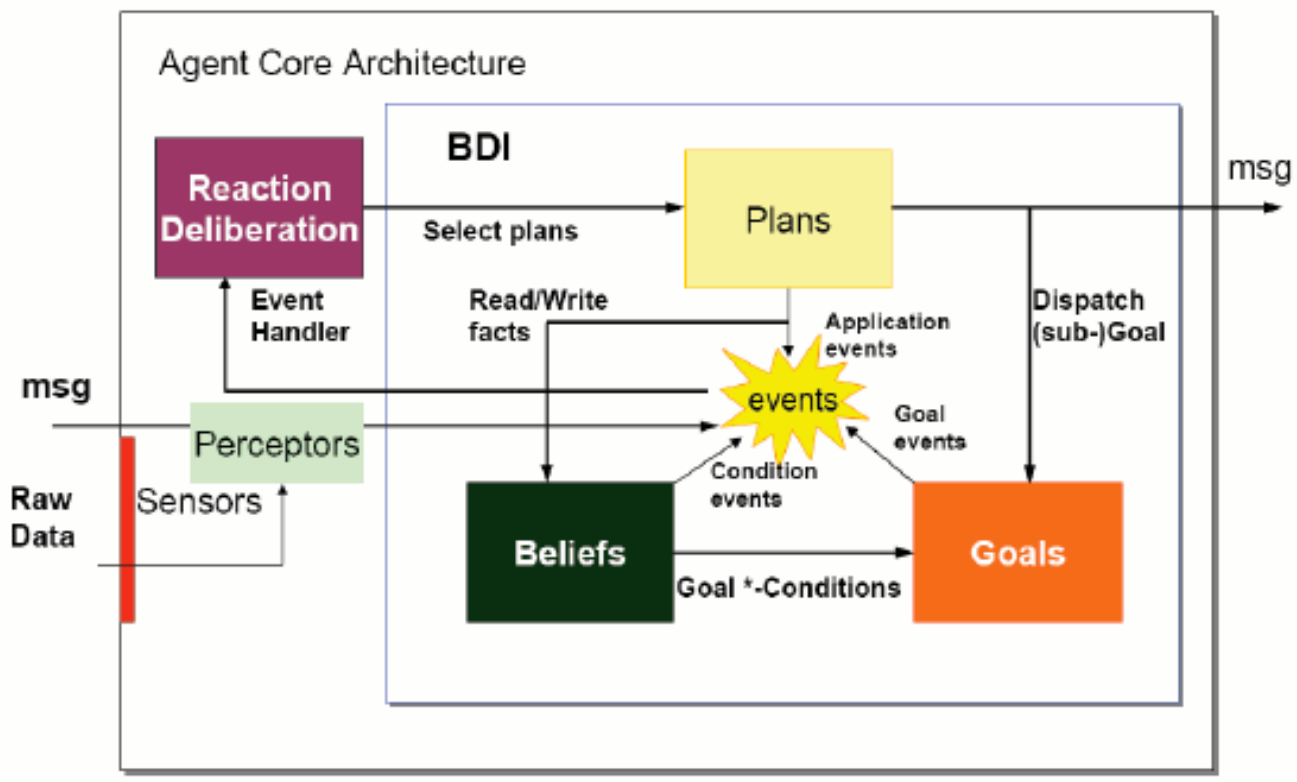


Figure 26 Jadex overall architecture.

Jadex agents reacts to incoming messages and internal events, and deliberate about their goals. To handle messages and events, and to achieve its goals, the agent selects and executes plans. The current beliefs influence the deliberation process of the agent, and the plans may change the current beliefs while they are executed. Changed beliefs in turn may cause internal events, which may lead to the adoption of new goals and the execution of further plans.

Jadex developed an higher level, adaptable, *deliberation strategy* shifting deliberation policies from application to architecture level. Systematic information and events establish dynamically, transparently to the agent programmer, specific interrelationships between Goals. The system automatically detects interdependencies at runtime reorganizing on the fly goal priorities and preserving a consistent mental state. The mechanism is realized through the management of fully dynamic network of inhibition arcs between agent's Goal, where instances of each Goal is limited

by an upper bound cardinality value (Pokahr *et al.* 2005a and 2005b). In essence Inhibition arcs allows to define explicit negative contribution relationship between Goals.

3.5 Conclusions

ISTC-CNR has prepared the experimental setups, both real and simulated, in order to carry out the tasks related to the FINDING AND LOOKING FOR scenario. The setups are particularly suited to tackle the problem of building action repertoires and using them to produce more complex behaviours. The hardware setup is based on a robotic Pioneer arm controlled by a computer and by prototypes of humanoid robots made available by the EU funded project RobotCub at Genova University (a specific research collaboration has been started with this University). The simulated setup is based on two 2D and 3D customized C++ simulators that allow very fast simulations, particularly useful for simulations that involve computationally heavy learning processes.

ISTC-CNR has also done all preparatory work to start with experiments in the two tasks of the GUARDS AND THIEVES scenario. The architecture suited for the first task is composed of three parts, and it is intended to address three tasks related to action control and its shift between the levels. The methodology is simulative, but the constraints given by the simulator (e.g. realistic physics based on ODE) make the results suitable for real robotic domains. The architecture suited for the second tasks is focused on the role of expectations in higher levels of cognition such as deliberative reflection; the simulation setup is ready to begin experimentations.

4 LUCS

ANTICIPATION IN A DYNAMIC WORLD

- ***The fish catching game (Game Room)***

In the fish catching game, the movement of the targets is very regular but there are two types of predictions that can be made:

- *the path of the fish*
- *the time when it will open its mouth.*

When the scene is viewed from different angles, the system need to predict the movements of the fish regardless of from where it is looking at it. Ideally, the learned model should allow for quick relearning (or reparameterization) when the viewing angle changes.

- ***The marble run game (Game Room)***

In the marble run, the movement is again very regular, but the different components of the game can be rearranged to produce different paths for the marble. These scenes combine the continuous dynamics of the ball with a compositional structure. This allows for generalization between different configurations of the elements of the run.

- ***Learning the two games at the same time (Game Room)***

To add some complexity to the previous tasks, the cognitive system could simultaneously look at and learn the different games. This makes the learning context sensitive. It also makes it possible to study how the current game can be used to prime the relevant features of the visual scene that should be used for anticipation. Ideally, the system should learn that there are two different games by itself by detecting the relevant contexts. The only given goal of the system will be to anticipate the state (e. g. location and velocity) of some predefined objects in the scene. By simulating a delay in the perceptual system (as would result if a robot was used), it becomes necessary to predict the behaviour of the moving object for tracking to occur.

GUARDS AND THIEVES

- ***Conflict in the access to valuables - complex (House)***

This is a social task involving several agents – several thieves and a guard. The session ends either when all the valuables have been collected or found (no matter by whom) or when the guard has arrested (caught) all the thieves as described in this scenario.

4.1 Robots

For the game room scenario with several interacting robots, LUCS has built six robots that can be controlled through Bluetooth from a remote computer (Figure 27a). An overhead camera is used to track the positions of the individual robots (Figure 27b). Obstacles (bricks) and other objects can be placed within the game area to form rooms when necessary.

For the control of the robots, a basic set of real-time path planning abilities have been implemented using a hybrid method based on A* in a grid representations combined with Bezier spline representations of paths. The method will be used as basis for anticipatory navigation through the environment and different levels of reactive, planning and anticipatory abilities can be switched on or off.

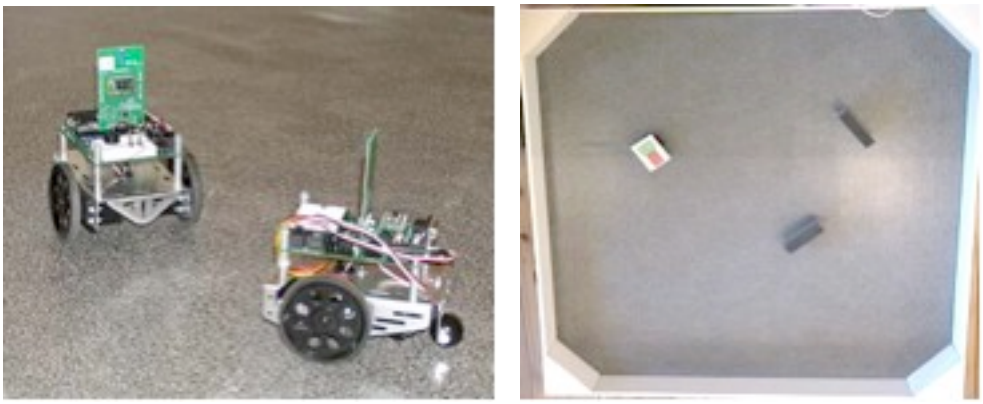


Figure 27 (a) The robots used for the game room scenario. (b) The overhead camera view of the room environment before processing. There is one robot and two obstacles present.

In addition, LUCS has developed a 2D simulator for the environment that is used for off-line testing of algorithms. This simulator will also be used by the robots to simulate the behavior of other robots during navigation and planning.

To identify the positions and orientations of the individual robots, the overhead camera (Figure 28) delivers a continuous stream of jpeg encoded images over TCP/IP. The image of the environment is first transformed into a rectangular shape and then color corrected before the robots and obstacles are identified. The environment itself is represented in a grid while the robots are tracked to continuous locations and orientations. The amount of information that is available to each robot can be controlled to simulate a limited field of view.

Figure 28 The AXIS 2130PTZ camera used for visual tracking of the robots in the game room scenario and for active tracking of moving marbles and fishes in the marble and fish games.

In addition to the six small robots, LUCS has designed a prototype for a larger (but still small) robot (Figure 29). This robot will eventually be used to carry an active camera system, which will be used to track targets in a dynamical environment. The advantage of using a robot over a static camera is that the effects of self-motion and visual perspective must be handled as well as the dynamics of the environment. The robot will also be equipped with a simple arm to allow it to interact with moving objects such as a ball or the fished in the fish game. The robot uses Mac Mini as its embedded computer and uses Ikaros (see below) for control and communication with a remote computer through wireless network.

Figure 29 The MiniBot, a prototype mobile robot which will be used to move the camera while it is tracking dynamic scenes like the fish game and marble game.

4.2 Video Recordings

LUCS has recorded video clips of several classes of dynamical scenes. Movies of the two scenes (fish, Figure 30, and marbles, Figure 31) have been recorded from five different angles and coded in a number or formats: (1) MPEG at a resolution of 640x480 pixels, 25 frames per second. This is the raw format to use when the complete visual recognition and anticipation task is addressed. (2) MPEG at a lower resolution of 320x240 pixels, 5 frames per second. This format is used as reference for the raw tracking data when a lower bit rate is desired. (3) Raw tracking data (coordinates and state) for the target object in each movie at 25 values per second together with a static description of the scene.

Fish Game Data

The raw tracking data for the fish game consist of the x and y coordinate of the target fish in the image and a third component that identifies whether the mouth of the fish is open. This data are be coded in ASCII files with four columns of numerical data. Since the scene is cyclic, then each data file contains one cycle with typically lasts less than 5 seconds. To allow generalization between different views the scene has been recorded five times from different viewing angles. The location of the camera relative to the center of the game is also supplied.

Marbe Run Data

The raw tracking data for the marble game consists of the x and y coordinate of the marble in the image and third component that identifies whether the mouth of the fish is open. This data are coded in ASCII files with four columns of numerical data. The tracked data contains the position of the marble from the time that it enters the scene until it disappears.

An additional file will contain a description of the elements of the scene separate from the position of the marble. The two coordinates for each element indicate the start and end of the marble run through the element. For elements without clear locations of this kind, both coordinates code the center of the element

There are five different scenes with different arrangements of the elements. Two of these scenes contains elements that partially occludes the pathway of the marble. Each scene has been recorded from five different visual angles. The location of the camera relative to the center of the game is also supplied.

Figure 30 Still image from the fish video clip.

Figure 31 Different simple marble run games with the same elements arranged into different dynamical scenes. The rightmost image shows the detected anchor points in the image that are used for the scene description.

4.3 Software Architecture

All implementations are done using the Ikaros framework (http://www.lucs.lu.se/Ikaros). The main components of the Ikaros systems are: (1) A platform independent simulation kernel currently running under Window, Linux and Mac OS X, (2) A set of computational modules implementing

different cognitive mechanisms or algorithms, (3) a set of I/O modules for interfacing with data files and peripheral such as robots or video cameras, (4) tools for building systems of interconnected models specified using an XML-based format, (5) a plug-in architecture that allows new models to be easily added to the system.

The system makes it easy to develop cognitive components that can be used in many different models and is thus ideal for comparison between different cognitive architectures or different combinations of cognitive modules. It also makes it possible to use many visual processing modules that have already been developed for Ikaros. There are currently over 100 modules in the system ranging from I/O to learning and perception. The adoption of several web protocols makes it easy for Ikaros to communicate also with other implementations of cognitive architectures in a client-server setting.

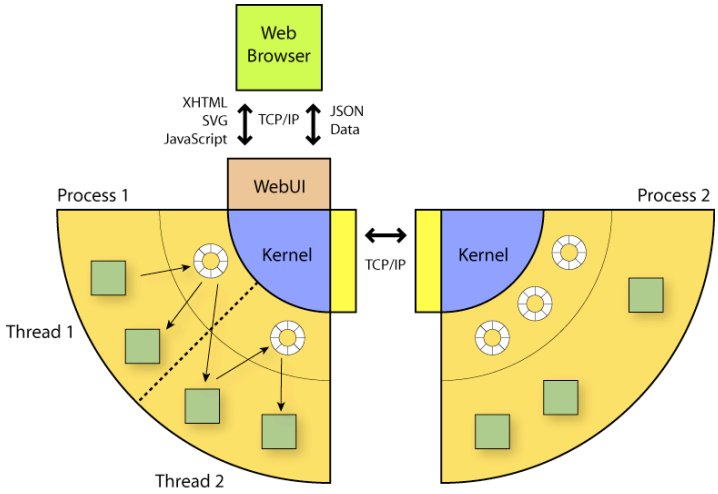


Figure 32 Overview of the Ikaros system. The system allows flexible development of complex cognitive systems of interacting modules implementing different mechanism (dark green). The processes can be monitored from the web based interface using a browser (light green), which communicates through a WebUI module (red). The kernel (blue) controls the execution of the individual modules and the communication between them. Several Ikaros processes, possibly running on different computers, can communicate over TCP/IP.

Ikaros has been extended with a web server that interacts with a web browser to show the state of a running Ikaros process (Figure 33 and Figure 34). The browser side of the viewer combines JavaScript and CSS with SVG rendering of images and graphs. A plug in interface similar to that used for Ikaros modules has been developed that allows arbitrary visual elements specified in SVG. The web view interface is compatible with Firefox 1.5, Camino and browsers using Adobe SVG 3.0 plugin (e.g. Internet Explorer and Safari).

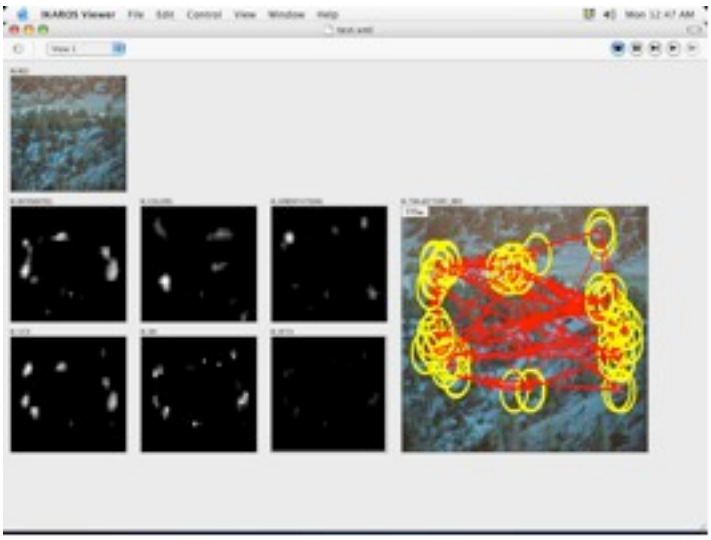


Figure 33 The attention model of Itti & Koch running in Ikaros. The visualization is shown in the Web client running on Mac OS X

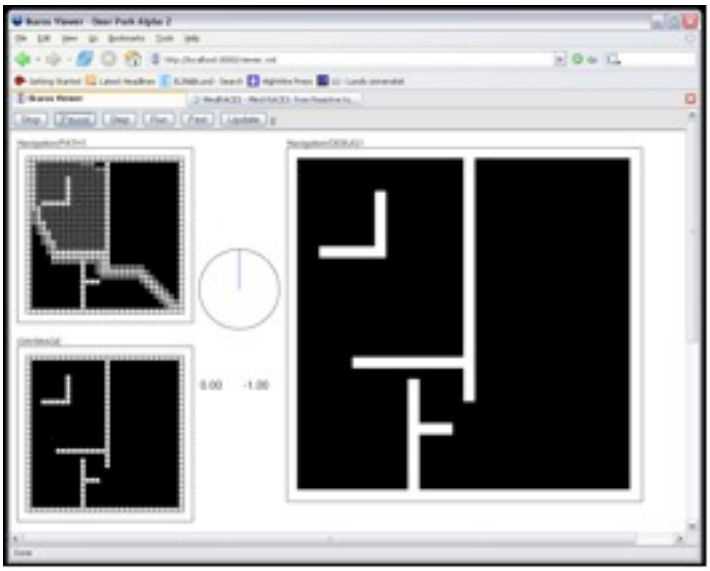


Figure 34 The path planning algorithms for the game room robots visualized in the Ikaros Web client running in Deer Park Alpha on Windows.

5 NBU

FINDING AND LOOKING FOR

- ***Finding a specific object (Game Room)***

The purpose of this task is to find a specific object in the environment (e.g. a red cube). The degree of detail in the description must be sufficient to define unambiguously a single object, not a class of similar ones. For example "red cube" is to be used in the case when there is a single red cube, and "big red cube" if there are several red cubes with different sizes and only one of them is big.

- ***Finding members of a class of objects by class description (Game Room)***

The purpose of this task is to find any object matching some general or partial description (for example "find a cube" or "find a red object"). As in the previous case, prediction or anticipation can be based on previous experience, recurring spatial relations, etc.

- ***Looking for an object in the House (House)***

This task is placed in the House environment. Coloured light signals might be positioned above/aside passages between rooms. These lights signal if the passage is open or closed, and might have periodic behaviours. In this task the robot's goal is to find an object that is hidden in one of the rooms in the shortest time or using the shortest way. The level of detail in the object's description may vary. In the case of class-definitions of the target, the purpose of the robot is to find any object that matches the given description.

In some conditions, the target's location is probabilistically biased towards certain locations (e.g. red cubes tend to stand on yellow cubes, although not always, or to stay in some rooms).

GUARDS AND THIEVES

- ***Conflict in accessing the valuables - simple (House)***

This task involves two agents – one thief and one guard. In the beginning several valuables are hidden in at least two different places or there are several accesses to the hidden place, in order to make the guard's task non-trivial. The session ends either when the thief has collected or found all the valuables or when the guard has arrested the thief either by blocking him or by touching him.

- ***Conflict in the access to valuables - complex (House)***

This is a social task involving several agents – several thieves and a guard. The session ends either when all the valuables have been collected or found (no matter by whom) or when the guard has arrested (caught) all the thieves as described in this scenario.
In addition to all the problems listed before this task implies that the thieves should be able to distinguish between guards (danger) and rivals/fellows (competition/cooperation).

- ***Coordination in accessing the valuables - several thieves (House)***

This is a social task involving several agents – several thieves (at least two). Some (types of)

objects are considered to be valuable and each player aims to find them all. Thus the participants have to play the roles both of the thief and the guard from previous tasks. If one thief blocks another (the way the guardian could block the thief) the first takes the valuable from the second if currently it is carrying any. The session ends once a player has collected/found all the valuables or after some fixed amount of time.

NBU approaches the tasks by creating agents that are capable of anticipation by making analogies with previously experienced episodes. For example, the agent could look for the hidden object in places where it was hidden in analogous situations, or in places analogous to the ones where analogous objects were hidden on previous occasions. Moreover, the analogies might be with respect to the objective spatial configuration of objects and rooms, or with respect to another agent's (a guard's or a thief's) previous behaviour, e.g. agent1 has previously hidden the bone behind the door of room1, and agent2 has hidden it behind the left most cube in the corner of room2, now being in room3 and knowing that agent2 has hidden the bone, our robot could anticipate by analogy that the bone is behind the left most ball in the corner of that room.

For implementing such a robot capabilities NBU will use the DUAL cognitive architecture and the AMBR model of analogy-making developed on its bases. The first step will be to use a simulated (virtual) environment (WEBOTS) to model the behaviour of the agents and the second one will be to use real robots (AIBO and Pioneer).

5.1 Physical environment

The scenario will be implemented by using AIBO robots (see Figure 35a) and/or Pioneer 3DX robots (see Figure 35b) controlled via wireless (WiFi) network remotely by a computer running all the required modules (see Figure 39). Thus the required processing power will be offloaded from the robot.

For the first task the hidden objects will most frequently be bones, but could be any other object. The objects in the room will be cubes and balls of various colours and the bone could be hidden behind any of them (see Figure 36). The AIBO dogs will go behind the corresponding object and if the bone is there they will collect it, otherwise they will fail or continue to search. When the Pioneer robot is used it will be able to grasp the corresponding front object with its arm and thus make the hidden object visible (if the anticipated position is correct). For the more complicated tasks the environment will be enriched by having separate rooms, doors between the rooms, light indicators on the doors (or near them), etc. or a complex labyrinth will be built.

Before running the actual robots, a simulated robot in a simulated environment will be played (see Figure 37). This will make the step of extraction of the data from the environment easier as well the control and monitoring of the robots actions. The NBU team will use the Webots environment for that purpose. This step will allow the team to directly run into the hard problems of anticipation using analogy-making, while the difficult tasks of perception of the real environment and acting over it will be simplified in this first phase using the virtual environment, while the tasks of perception and manipulation of the real physical environment will be left for the second phase of

the project (as planned). This will ensure a gradual development process and eliminate high-risk pathways.

Figure 35 (a) NBU's dogs (AIBO ERS7) playing with a bone and (b) NBU's Pioneer 3DX.

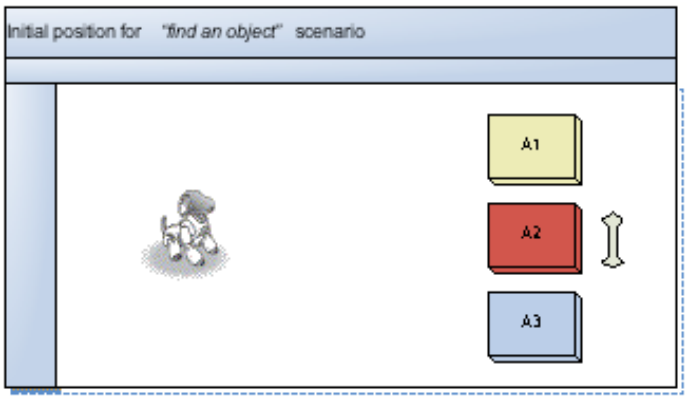


Figure 36 The robot in its environment with a hidden object.

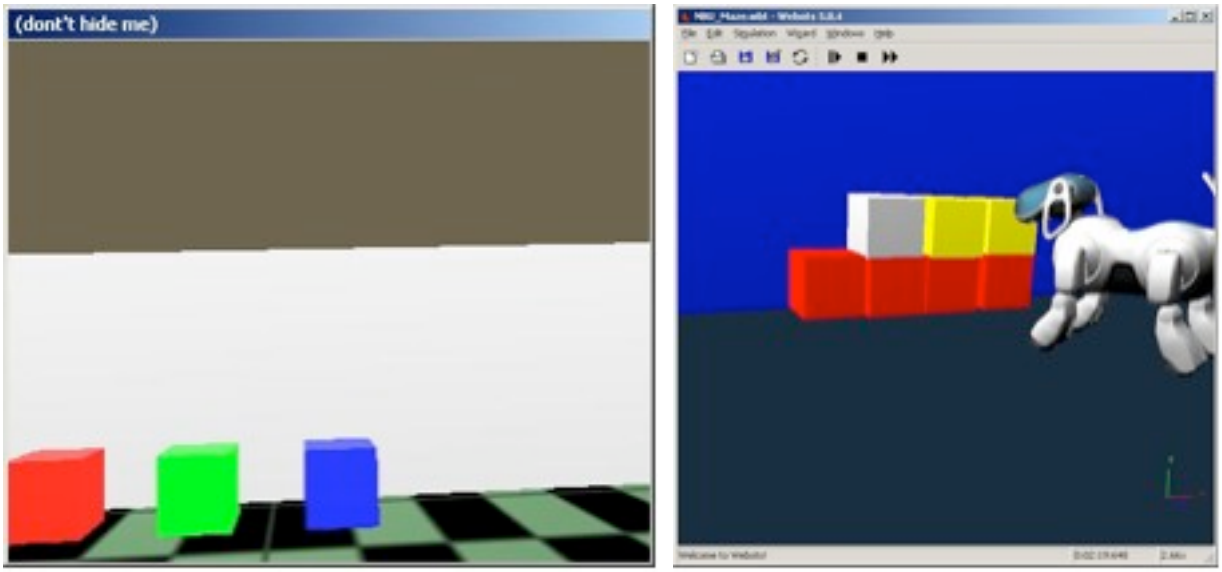


Figure 37 (a) 2D image from a simulated AIBO's point of view (Webots) and (b) external view of a simulated scene (Webots)

5.2 NBU's system architecture

Solving any of the tasks in the scenarios will require perceiving the scene and building internal structural representations, retrieving analogous situations from episodic memory, mapping the current situation onto the retrieved one, transfer of a prediction and/or a plan for action from the old episode to the new one, evaluation of the transferred knowledge, actually performing the plan with physical actions, and possibly learning (e.g. generalizing the episodes and building schemas). This cycle may be grouped into three subprocesses: representation-building, reasoning by analogy, and performing actions (see Figure 38).

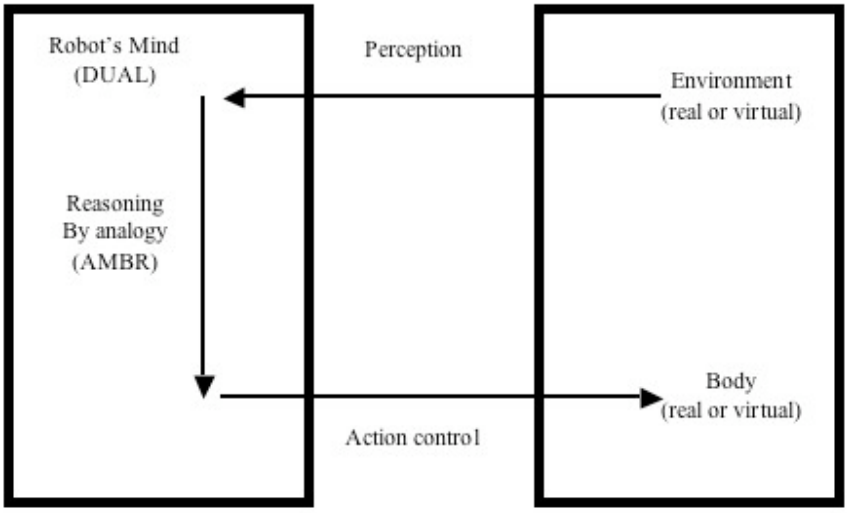


Figure 38 Basic processes required for solving a task in the environment.

Thus the system to be implemented by NBU will be organized in a three-tier fashion. Each layer will be implemented by different independent modules. This allows that any of the three tiers can be upgraded or replaced as requirements or technology change, in this way limiting the impact on the others parts of the system. The different tiers and their interaction are shown on the next illustration (see Figure 39).

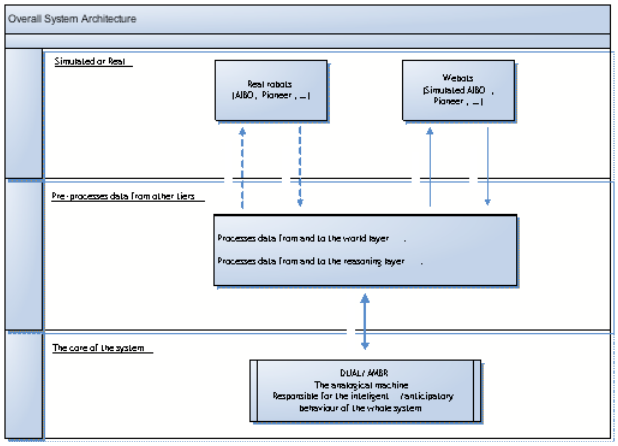


Figure 39 NBU's overall architecture.

5.2.1 The world layer

It can be either simulated, using appropriate software like Webots, or realized with a real robot living in a real world environment. In the first step NBU will focus on the simulated world approach as it will facilitate the retrieving of structured data (objects and their relations) directly from the environment. The description of the next layers will focus only on a simulated environment case, the real world scenario will be implemented only after the simulated one is successfully developed.

The NBU team has purchased the AIBO robots and Pioneer robot and started learning them and experimenting simple programming. It also purchased the WEBOTS environment and started learning and using it as well as experimenting the relations between the real and the simulation robots. Webots is a professional mobile robot simulation software, which allows the simulation of physics properties such as mass repartition, joints, friction etc. It is possible to control the robot by setting the position of its body parts (with regard to the joints), as well as to obtain information from the robot's sensors. Each simulated robot can be modeled individually. Webots comes with several models of real robots, like AIBO and Pioneer. This enables transferring tested behaviour to real robots. In addition, Webots allows connection with external software modules.
The results so far include:

- creation of worlds with physical laws
- creation of object with any shape
- creation of robots and endowing them with behaviour (i.e. programming their behaviour)

5.2.2 Middle layer

The general purpose of the middle layer is to serve as a mediator between the other two levels, effectively translating and filtering out the information from one layer to the other. More precisely, given the description of the world obtained from the *world layer*, the mediator should filter out all the information that cannot possibly be perceived by the robot (given its current position and the direction it is facing, its sensors etc.) thus creating a reduced scene representation. Next, the needed relations have to be extracted for the objects in the reduced scene representation. At the final step all this information has to be represented in a suitable form and sent to the next layer.
For the inverse operation, the data from the reasoning layer (the plan) is transformed into sequences of low level commands to be sent to the simulator for actions to be carried out.

5.2.3 Reasoning layer

The reasoning layer is where all the information coming from the middle layer is processed and high level commands are issued backward to it. Here is where the anticipation is built based on the description of the current situation and knowledge of past situations in memory.

The communication between the reasoning layer and the middle layer is by data exchange, the data being in XML format. Each exchange should be done with a complete XML file conforming to the DUAL/AMBR XML Schema. This standardized way of communication should permit DUAL to be easily interfaced in different ways ranging from direct access (like web services) to other models/layers. This layer is in fact DUAL.

5.3 DUAL

DUAL is a general cognitive architecture developed at NBU which supports emergent computations based on the combined behavior of many micro-agents. Representation of knowledge is decentralized and distributed over coalitions of micro-agents. The architecture is hybrid and each micro-agent combines symbolic processing with connectionist spreading activation. The individual speed of symbolic processing of a micro-agent depends on the dynamically computed activation level of the same agent. Thus the two aspects are highly integrated. A model of analogy-making, AMBR, was developed based on the DUAL cognitive architecture. AMBR will actually do the reasoning needed for predicting and anticipation in the selected scenarios.

The problem is that the current realization of DUAL is done in Alegro Common LISP. This version of the architecture is a research product and is not ready for direct use in applications such as real robots. To overcome these limitations a very demanding task has been undertaken, namely to reimplement DUAL in C# in a UNIX environment. This language was selected to ensure easy interface the C# version of DUAL with various applications which will be needed for fully controlling the robot. The interface will be implemented either by using the .NET build mechanisms combined with a .NET languages or by using a standardized way of communication such as XML messages or Web Services.
This will permit easier creation of extensions for the model that will preprocess the data i.e. creation a sort of high level perceptual level (the middle layer of the system architecture can be regarded as this kind of extension)

Running DUAL on .NET will also permit using it on other computer platforms (using Microsoft .NET or Mono).
New functionalities will be developed within the new re-implemented model of DUAL, some of them are: automatic episode visualization using graphs, easier episode creation, management of the knowledge bases (general and episodic ones), real time visualization of the internal mechanisms of DUAL (visualization of the thinking process).

5.4 Acquaintance with the Robots and the Simulator

Different tasks are being carried on in order to gain in-depth knowledge of the way the robots can be controlled and simulated using Webots.

5.4.1 Making the simulated robot move realistic

On the basis of the analysis of the scenarios the following 'elementary' motions are implemented for the simulated AIBO:

- moving straight (both ahead and backwards) and aside (both left and right)
- turning (both left and right)
- manipulating objects (with mouth and/or paws)

Some motion classes are required for "long" movements from current to some other position. Others are for precise positioning, for example when the AIBO has to face (and may be take/bite) an interesting / valuable object.

The Sony tool MEdit (see Figure 40) has been used, which allows its users to construct AIBO positions (*.pse files) and to make animated smooth transitions (*.kmf files) across them. In addition this tool exports the motion animation files in format used by Webots - *.mtn. Although the animation may seem like an artificial object in the simulation, it affects and is affected by all the objects in Webots environment as expected (kicking, crashing in an obstacle, etc).

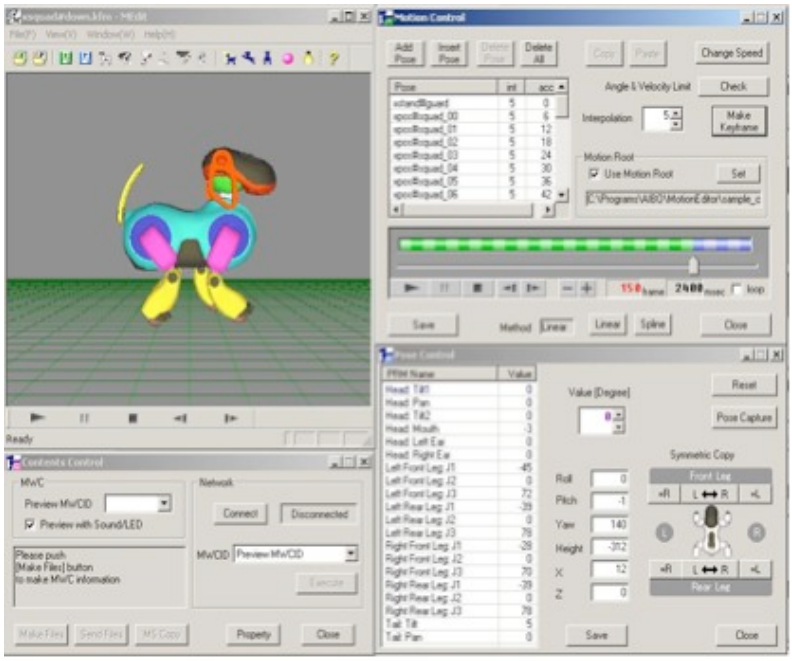


Figure 40 Sony MEdit tool in action.

The AIBO skeleton pose is completely defined by its 20 joints' angles. Thus any position imposes some dependencies among those angles. NBU has developed a tool which calculates these joint angles according to the required position and moreover, computes all the positions for a motion (the more the positions are, the more adequate the motion is – because the angle speeds for various joints are different, it may happen that during the animated motion the paw "sinks" into the floor).

Later this problem has been solved more effectively by directly supplying the joint angles values to the simulated AIBO. Although it requires many more calculations (the correspondence to frames in animated motions has to be calculated by this algorithm), it makes no distinction between rough and precise motions and in this way the motion becomes more purposeful (not a sequence of strange movements as in an army parade).

These algorithms (both approaches) are transferable to the real AIBO robot, thus serving a double purpose.

5.4.2 Create a simulated vision system

The idea is to use the environment and robot representation in Webots and generate a symbolic description of what is perceived by the robot. The intention is not to have an image-recognition system, but to use the simulation environment parameters and to construct with them the picture the simulated robot should be seeing. In this way, the robot will have a mediated perception which supplies the robot DUAL/AMBR model with an adequate (symbolic) description of the scene which is categorized in terms of the concepts of the robot.

The solution of this problem is related to the solution of two sub-problems, which are described in the following sub-sections.

5.4.3 Identify the state of the current environment

This means obtaining the current position and orientation information for all the objects in the current environment.

5.4.4 Filter the visible information

This includes the ability "to see" only objects in current visual cone. In addition to the primary calculation (whether the intersection between the object's shape and the visual cone is not empty), this requires the problem of doing some more advanced calculations to eliminate the visually overlapping objects (entirely behind an obstacle) and the problem of recognizing partially visible objects. Because the Webots does not provide such functionality we have to handle this on our own.

No sub problem has a priority in solving (it is a waste of efforts to have positions of the objects which are not visible, and yet no decision about which objects are seen can be made without knowing their positions) thus some generic solution should be found.

And finally, the visible portion of the world has to be symbolically reported to the middle layer.

5.4.5 Establish a two-way connection with external software module

The external software module we are interested in is the middle layer. The sensory system (visual and later on, acoustic) should supply the information about its neighbouring environment, and conversely – the middle layer should be able to command the Webots simulation at any moment (this is one of the reasons why we decided to solve the movement problem directly and abandoned the motion animations – the body could not do anything observing the law of inertia until the entire animation cycle is completed). Webots enables TCP/IP connections in both directions.

The middle layer will be created to control the robot's actions and to supply the perceived data to DUAL/ AMBR in a suitable form. Separating it in an additional level is needed to have the robot level isolated – i.e. whether it is an AIBO, a Pioneer or Webots' simulation of any of them, the communication will not be affected. Both systems will communicate through high level interfaces. Here is a list of methods the control interface **IRobotControl** will provide:

Method	Parameters	Description
LookIn	direction	Makes the robot look in direction requested
LookTo	position	Makes the robot look towards the position requested
GoTo	position	Makes the robot go to position requested
Stay		Makes the robot stand still
Turn	angle	Makes the robot turn in angle in radians requested
Take	object	Makes the robot take the object requested (the object contains information about its position)
PlaceIt	position	Makes the robot place the object it is carrying into position requested

Table 3 Robot controlling interface.

In all the methods listed both direction and position parameters could be relative (compared to its current sight direction or its current body position) specified or absolute (according to the scene map).

IRobotControl interface will have as many implementations as robots (in our case four – AIBO and Pioneer). In addition, for simulations an extra interface shall be provided to serve as delegate for IRobotControl and to supply commands to Webots' world robots.

The feedback interface will supply robot sensory information to the middle layer in XML.

5.4.6 Make the simulated robot solve some base problems

The simulated robot described so far has several abilities but all of them may be classified as inborn. This point regards endowing the robot with some instincts, i.e. it concerns problems the analogy-making system AMBR will not take care of, like:

- Moving from current position to a desired one
- Object manipulation
  Both of them should use solutions 3.1 and 3.2. In addition, the optimal (according to criteria supplied) motions have to be chosen.

- Searching
  There are two kinds of searching – visual and walking. Again solutions 3.2 (and 3.1) should be involved and if the description is fuzzy/incomplete probably some consultations with the "mind" could be done, i.e. 3.3.

- Notification
  The pre-mind system discussed here should be programmable by the external superstructure to perceive important objects/actions/events and to notify this superstructure about them. It has also to filter the useless information before supplying the perceived world description to the mind.

Two examples:

- in single scenarios if the task is "find a big red object", the shape information might be considered useless by AMBR and thus not transferred
- in social scenarios it is vitally important for the "bad guys" to notify opponents/enemies

5.5 Knowledge representation and management

One possible initial state for the task "*find an object"* is shown in Figure 36.

In order to perform the task there are several states that the simulation must achieve as shown in Figure 41. The states are depicted as a boxes connected with a solid arrow. After the initial state follows the intermediate state, in order to achieve the desired state (intermediate state at this step), the robot must perform an action that should cause the state transition (in this case the action is *find the bone* which in turn must e achieved by moving to one of the three possible positions A1, A2, A3)

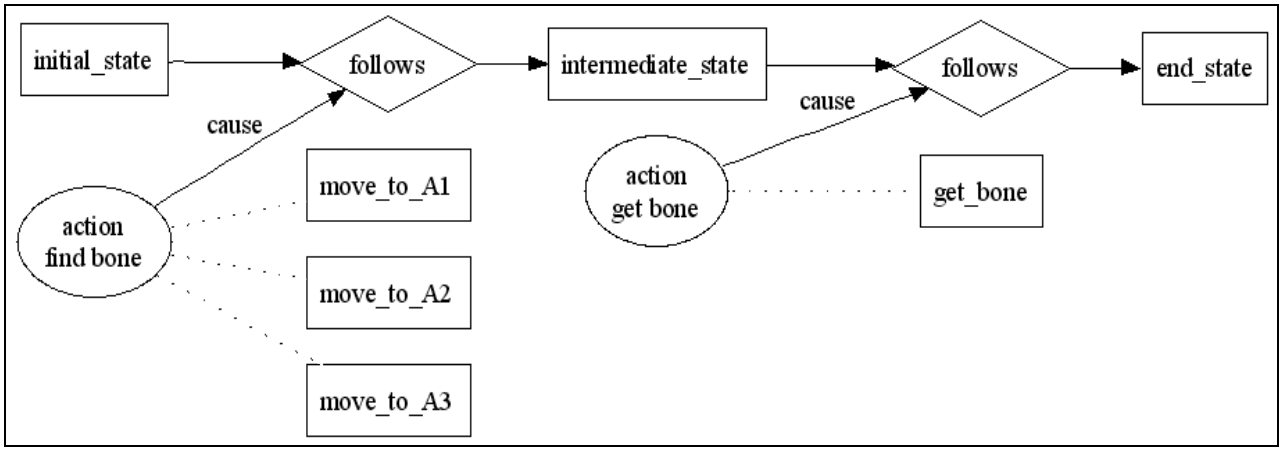


Figure 41 The different states that must be achieved in order to correctly solve the task.

The detailed representation of the initial state is shown in Figure 42.

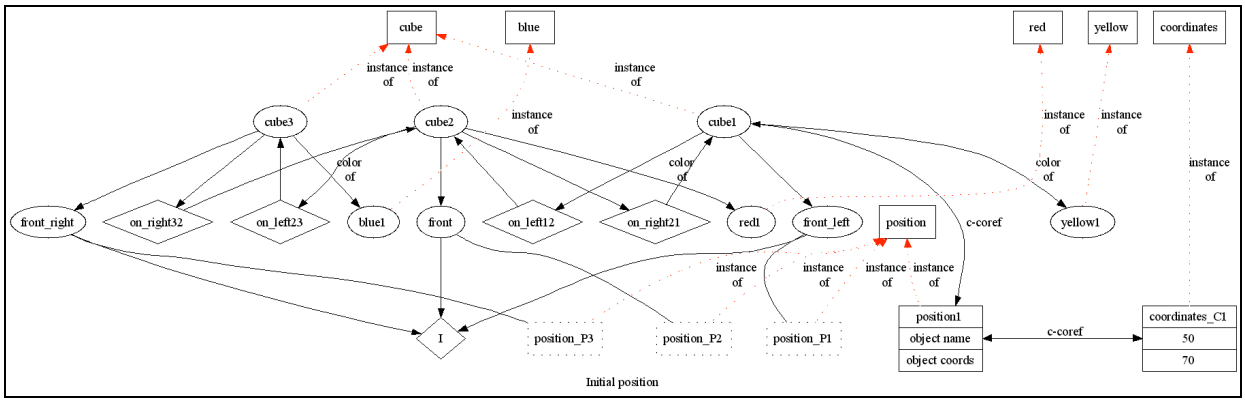


Figure 42 Detailed representation of the initial state for the task "find an object" in DUAL/AMBR (generated automatically).

5.5.1 Development of knowledge representation in DUAL/AMBR

Entities to be represented	Use in the scenarios	Availability status
1. Concepts and instances	everywhere	available
2. Relations and propositions	everywhere	available
3. Episodes, events and states	everywhere	present
4. Goals	everywhere	present
5. Absolute positions	everywhere	missing
6. Motor actions	everywhere	missing

Table 4 Scenario elements representation.

- **Representation of Concepts and Instances**
  The concept-agents represent classes of objects, whereas the instance-agents represent individual entities. The concepts are arranged in a taxonomic semantic network via :subc and :superc links. The instance-agents are related to the concepts via links :inst-of and :instance. Each concept can be connected to zero, one, or more of its super-classes or sub-classes. It is an assumption, however, that the top-down links :superc and :instance represent only the most *salient* subclasses and instances, hence they cannot be too many (usually 1-4).

- **Representation of Relations and Propositions**
  Relations are represented in the same way as their arguments – via a network of concepts and instances. The conceptual co-references (c-coref) connect two complementary aspects of the same entity.

- **Representation of Episodes, Events and States**
  Episodes, events, and states are represented via networks of agents, organized around the following principles (see Figure 43):
  - Each *event* represents a certain change in the environment or in the current goals. One agent serves as a head of the event, a second agent states for its *initial state*, and a third one - for its *final state*. It is possible the events to have also one or more *intermediate states*.
  - The *states* are representation of the entities that are relevant to the respective event, i.e., the entities that change, together with the reasons that cause this change. Thus, each state consists of a head (shown on the picture) and a coalition of agents – objects, relations, etc.

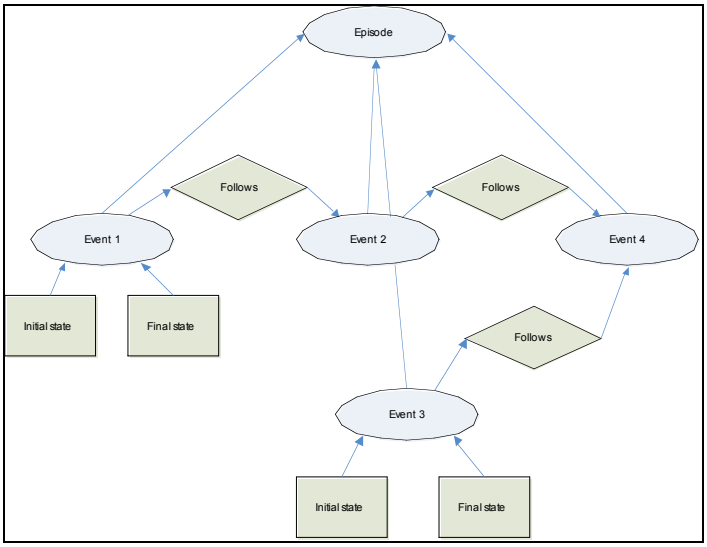


Figure 43

- The *episodes* represent larger pieces of knowledge and usually consist of several events. For example, one episode could be the story when the robot searched for a red cube in a labyrinth. During this search, several events have been happened. Some of these events could be caused by the robot (moving from one room to another or from one place to another; manipulations on the objects, etc.), some of them – by other subjects (other robots could manipulate the objects or to change their positions; external events like change of the light system may occur).

Some of the entities could by invariant during the events, but relevant for the whole episode. In such a case, they point only to the head of the episode, not to the head of the all events. The same principle is valid for the entities that are relevant to the events but are invariant to the states, included in the respective event. These entities point only to the respective event, not to each single state.
The events could be interconnected each other with relations like *follow* and *later*. However, this is not necessary – the exact order of the events can be forgotten.

- **Representation of Goals**

The goals are actually states or events. In each task the robots have one main goal – to reach from the initial state to the goal state, or a certain event to happen. Thus, the representation of the goal state is the same like the representations of the other states and events. The difference is that the goals stay all the time attached to the GOAL node and supply the relevant to the respective goal entities with activation.
However, in order the global goal to be reached, it should be dynamically subdivided into smaller subgoals. Transfer mechanism is responsible for the creation of these subgoals; evaluation and decision mechanisms restrict them.
Each change of the subgoals actually is an event, even if nothing changes in the environment.

- **Representation of Absolute Positions**
  It is possible that some slots of certain agents are filled with numbers that represent absolute positions. Later these numbers can be used for performing calculations for estimating other positions or for planning certain actions to reach or to avoid the respective objects.

- **Representation of Motor Actions**
  Some large-scale movements could be represented in schemas. Examples for such movements are reaching an object, which the robot sees; movement to a certain absolute position; movement to a door and entering in the neighbor room, etc. Motor actions can be represented as single entities, but also as parts of larger condition-action-prediction schemas.

5.5.2 Learning

During perception and actions large temporal KB is created. After decay, some of the temporal agents and links should disappear, but the most relevant (active) ones should change their status into permanent and should stay in the LTM.
Probably, the permanent links also should change their weights, depending on their use.
Learning strongly depends on the mechanism for supplying the model with input and goal – sequential direct influence. The same is true for perception and selective attention – they influence directly and sequentially the learning.
On turn, learning influences retrieval and generalization directly and sequentially. Learning influences also decision-making indirectly.

5.5.3 Transfer

Some elements from the past – situations, single objects, relations, actions, etc. should be transferred in the new episode. Transfer mechanism is necessary to complete the analogy between two domains – to fill the missing relevant relations, objects, schemas, etc. in the current situation.
Transfer is strongly integrated with decision making. This integration is overlapping and a direct one. Actually, the GOAL/INPUT mechanism influences all other mechanisms because it determines the relevant items.
Transfer depends also on retrieval on generalization, on schemas with condition-action-prediction part. On turn, actions directly and parallel depend on transfer mechanism. Transfer determines the set of possible actions to be performed. On turn, the really possible actions influence decisions (particularly evaluation) and respectively - transfer.

5.5.4 Decision Making

Decision making involves several sub-mechanisms – evaluation of the transferred knowledge; decisions, concerning selective attention; decisions, concerning actions; decisions about subdivision of the goals.
The evaluation of the transferred knowledge is very important. For example, it is possible the system to transfer a solution from a past episode that involves a certain action. However, in the present situation this action can not be performed because of certain restrictions. In such a case, the model should decide to ignore this transfer and to pressure the system to search for other analogical situations. Thus, from one hand, micro-analogies between the elements from the current state and

different past episodes should produce novel solutions. From the other hand, the decision-making mechanism is responsible for choosing between two or more transferred condition-action schemas.
This sub-mechanism – evaluation of the transferred knowledge – is integrated with the transfer mechanism – parallel, direct, and one-mechanism based integration.
The other decision-making sub-mechanisms involve selective attention, actions, and division of the goals. All four sub-mechanisms are integrated in a single decision-making mechanism, which is directly integrated with retrieval, learning, action, transfer, selective attention, and emotions.
Decision-making influences also GOAL/INPUT indirectly, mediated by attention. On the opposite, the GOAL/INPUT mechanism influences directly decision-making, because the former determines the relevant items.

5.5.5 Generalization

Generalization is responsible for creation and management of the conceptual network, and also for creating the different types of schemas, in particular, schemas with condition-action-prediction part. Thus, we can separate generalization on a micro-level (single concepts) and on a macro-level (large schema coalitions).
Generalization consists of two integrated sub-mechanisms – automatic one and decision-making based one.
Automatically, each concept can manage its relevant instances, and if certain relevant information is common for all these instances, the respective concept should generalize it. However, it is necessary also each new instance that becomes relevant to be able to evaluate whether some facts contradict to the generalization. Thus, generalization could not be separate from the evaluation of the performed generalizations.
The generalizations on a micro-level can grow up, in order to satisfy structural consistency, and as a final result, large schemas to be generalized. In order to ensure such structural consistency, several agents should be monitored simultaneously. Analogically to the mechanisms for structural correspondence in analogy-making, this violates a little bit the architectural principles for local computations only. However, this price is not so large for the expected profit.
Generalization is strongly integrated with learning, attention, GOAL/INPUT mechanisms. Generalization sequentially influences retrieval, transfer, schemas. Generalization is sequentially influenced by attention and decision-making mechanisms.

5.6 Conclusion

The agents to be used in the scenario are a Pioneer 3 or/and an AIBO. The perceptual capabilities of the robots will be helped by a clear and simple coding of the objects and the relations among them (color and shape labels). The robot scans the scene and encodes it by using a conceptual system which is part of its long term memory and is organized in the form of general knowledge (e.g. cubes, obstacles, etc.) and of episodic knowledge (e.g. previous episodes of activities). The robot is given a goal or generates it. On the basis of the goal and of the perceived scene it retrieves relevant episodes from its episodic memory or general knowledge schemas and generates an action plan and expectations about the resulting situation. The actions are quite general e.g. 'go toward the red cube', 'take the object', etc. and the details of the safe execution of the action will be taken care of by a supervising module. This level of description has been adopted because of the difficulties that are foreseen to bridge low-level perceptual capabilities and the high-level semantic tree type of encoding of knowledge needed for DUAL/AMBR. This will allow exploring the full anticipatory potential of analogy by the NBU team.

The environment will be a standard office room but for the time being only simulated environments have been built using Webots. In this simulation environment worlds can be created involving objects with various shapes and colours. At the same time, robots can be incorporated in the simulations and endowed with behaviour. Webots possess several pre-programmed robots – Pioneer 3 and AIBO among them – which makes it convenient to be used in our case. NBU has focused on the simulation of AIBO robots in Webots so far.

6 OFAI

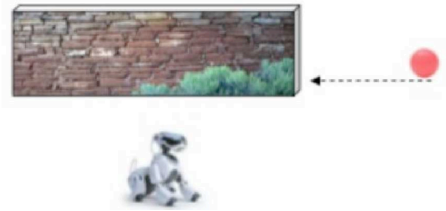
ANTICIPATION IN A DYNAMIC WORLD

- ***A "dog" growing up (Game Room scenario)***

This task is a bottom-up scenario, inspired by the idea of interactivism, tailored to implement and evaluate the artificial immune system architecture proposed by OFAI. Therefore the scenario is divided into three developmental stages:

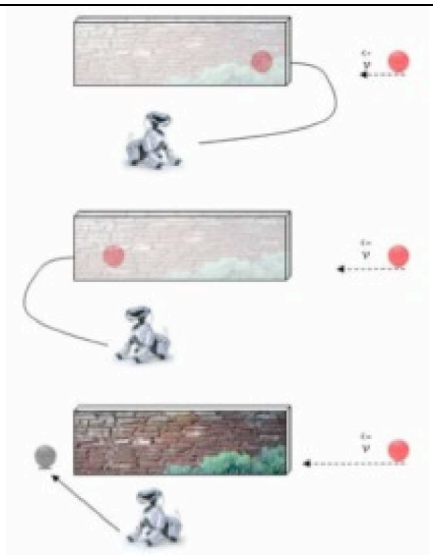
*In the **first, early developmental stage** ("capturing basic how-to knowledge"), the robot starts basic interactions with its environment (like walking around, looking at things, poking them), driven by basic instincts and motivations, capturing basic episodes of experience. After a specific amount of time and training the robot learns through reinforcement which interactions environmental features (or to be more specific objects) allow; thus in the first stage the robot acquires concepts of objects and the world itself, if not to say affordances of objects.*

- *Through interactions the robot gains knowledge about concepts of objects and the world around it.*
- *The main developmental achievement is to acquire object generalisations and certain concepts (one might even say affordances of objects such as the affordance of an object to be moved).*
- *As shown in the figure below, one achievement of this developmental stage will be e.g. to learn to expect where a moving ball will reappear after moving behind an obstacle (e.g. a wall).*



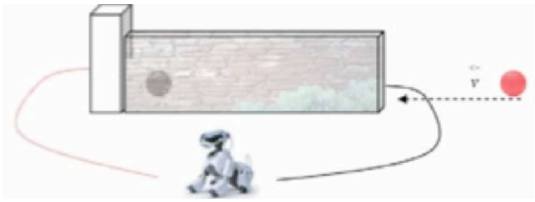
*In the **second developmental stage** (generalisation), after having acquired certain basic how-to knowledge about interacting with the world, the robot learns generalisation. The "how-to" knowledge which has been acquired in the first level is anchored in the following learning process.*

- *Now the robot should develop more sophisticated concepts, such as object continuity – e.g. it learns to anticipate that, depending on the speed, the ball will reappear on the other side of or remain behind the wall.*
- *If the ball moves behind the wall with a low speed, the robot should learn to look after the ball on the right side of the wall, if the speed vector increases it should learn to go look after ball on the other end of the wall.*



In another setup, the wall might be blocked at the end, and a ball, coming in with a high speed, normally reappearing on the other side, now does not reappear, and there is a sound (the ball bumping against the wall) coming from behind the wall.

- *In this case the robot should find out, that something has changed and that it needs to go and search for the ball from the right side again.*



*In the **final stage**, as the robot has seen the ball disappearing behind several "hiding places" (walls, obstacles), it shall now learn how to find the ball again and move around, looking for the ball, anticipating it to be in one of the observed "hiding places".*

- *The goal is to instantiate a hunter-prey sub-scenario, where prediction of behaviour, based on the capabilities acquired in stage one and two, is implemented.*

A second robot can be added to the scenario (i.e. KURT 3D, for further details next section).

This task will converge into the following scenario: by observing the prey, the AIBO should "hunt" the prey by simply intercepting it, or after some experience, going to a place where it anticipates the prey will go, realising an offensive tactic.

- *The robot thus performs epistemic actions such as "look if" or, "look for". This offers a bridge to constructive perception, what means that in general, what is seen is interpreted by the means of what is expected.*
- *Expectation then can lead to "asking questions to the world".*

6.1 OFAI scenario in detail

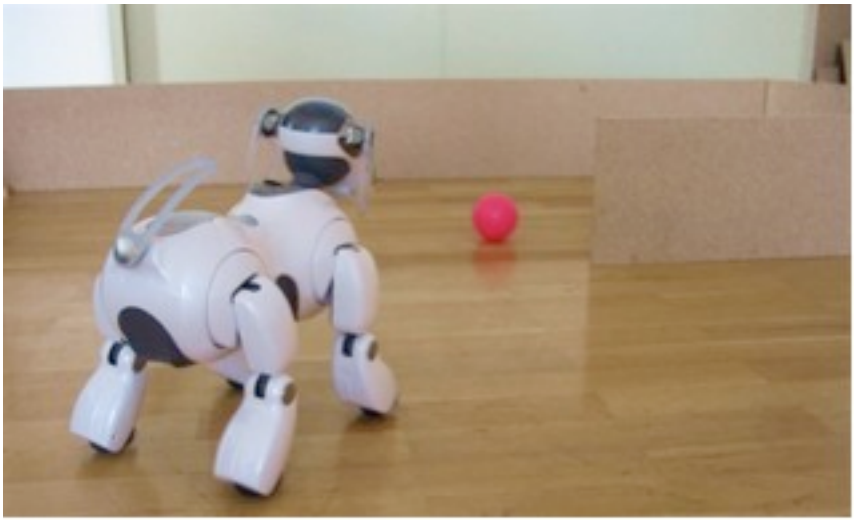


Figure 44: AIBO watching a ball disappear behind a wall

6.1.1 The OFAI test bed

The OFAI test bed consists of a fence-like barrier made from wooden boards. The exterior fencing is necessary to prevent the AIBO robot from taking French leave. The contents of the test bed vary according to the different developmental stages. In the first level inside the test bed, a series of boards will shape a small straight wall and balls will function as (moving) objects to interact with. Later on other objects such as balls in various shapes and salient colours, the IBone and building bricks will be introduced.

In the final stage the AIBO robot will share the test bed environment with a KURT 3D robot, implementing the hunter-prey scenario.

6.1.2 Scenario level one – capturing basic „how to" knowledge

In the first level the number of objects initially is limited and the objects themselves located in the test bed are deliberately kept simple. This facilitates developing a stable first artificial immune network level and allows starting tests without requiring all the programming of external filters in the first stage.

Objects: Balls (pink, white, etc., in any case in salient colours) and a wall (wooden boards aligned to a wall)
Tasks: Discover that there is something out there, and react to micro-genetic and developmental selection pressures[1].
Result: Develop a stable first level of the AIS.
Complexity:

- Simple object recognition
- Simple attention mechanisms
- Simple object tracking
- Basic movements
- Grasping concepts of the objects in the environment by interacting with them
- Learning to react to selection pressures[1]
- Developing basic behaviours

Requirements:

- Filter pool (basic colour tracking filters, rudimentary virtual sensors, edge detection filters, basic motion, etc.)
- Interfaces (communication interfaces, etc.)

Prediction/Anticipation: The artificial immune system architecture has innate predictive and anticipatory capabilities, realised through the threepart antibodies, consisting of a condition part *c*, action part *a* and expectation part *e*. At this level the robot develops more complex behaviours and concepts by evaluating the assumed expectation *e*, which results from performing action *a* under condition *c*.[2]

[1] (Bickhard 1999) Both, interactively and constructively, perceptually and developmentally, the fundamental form of influence from the environment to an individual is in terms of relevant selection pressures on the interactions, and, thus, the interactive organisations, of the system.
[2] For details see D4.1.

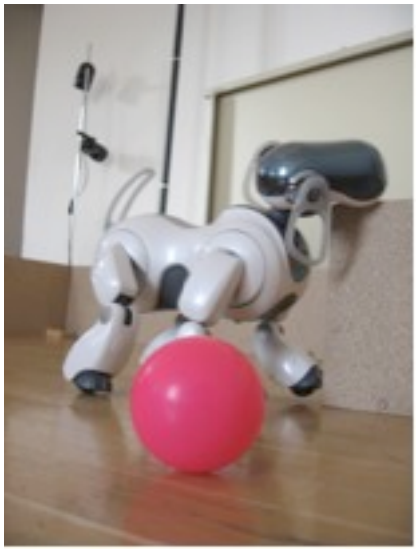


Figure 45: AIBO starting basic interactions with the environmental features, reacting to selection pressures.

6.1.3 Scenario level two – generalisation

In the second developmental stage, the "how-to" knowledge, captured by sequences of antibodies in the first layer, is anchored. Additionally this allows the robot to demonstrate the characteristic behaviour describing the interaction with a specific object, just by accessing the antibody sequence related to that object, as well as the creation of a topological map of the environment which corresponds to the topology of the second layer RLA network.

Thereby, as mentioned above, the robotic agent now develops more sophisticated behaviours and concepts, such as the concept of object continuity – e.g. it learns to anticipate that, depending on the speed, the ball will reappear on the other side of the wall, or remains behind it.

Objects: Different types of balls in various shapes and salient colours, the IBone and several toy building blocks.
Tasks: Develop more sophisticated concepts and behaviours.
Result: Generalisation of the first artificial immune system layer, by development of a stable second layer.
Complexity:

- Object recognition
- Attention mechanisms, selective attention (only the information relevant to object description should be processed , what is innate to the AIS)
- Anchoring objects and concepts acquired in the first developmental stage
- Anticipatory object tracking based on the concepts acquired in the first developmental stage
- Grasping of concepts, such as object continuity, etc.

Prediction/Anticipation: In the first developmental stage of the artificial immune system architecture, concepts about the nature of the environment will be formed by interacting with it. In the second scenario and developmental level, these concepts can be used to build hypotheses. Planning and anticipation now occurs as a dynamic cascade of internal events. For example, a goal, represented as an antigen, which is injected into the system.[3] E.g. in a situation, where the robot's goal is to seek a ball, hypotheses about the location could be tested by inserting antibodies into the AIS and then trying out the most promising hypothesis. When the robot anticipates the ball to be behind a wall, it will approach the anticipated location, where all pink objects will primarily focus the agent's attention (in the beginning the ball is the pink IBall, later on, as mentioned above, additional balls with different colours and shapes will be added).

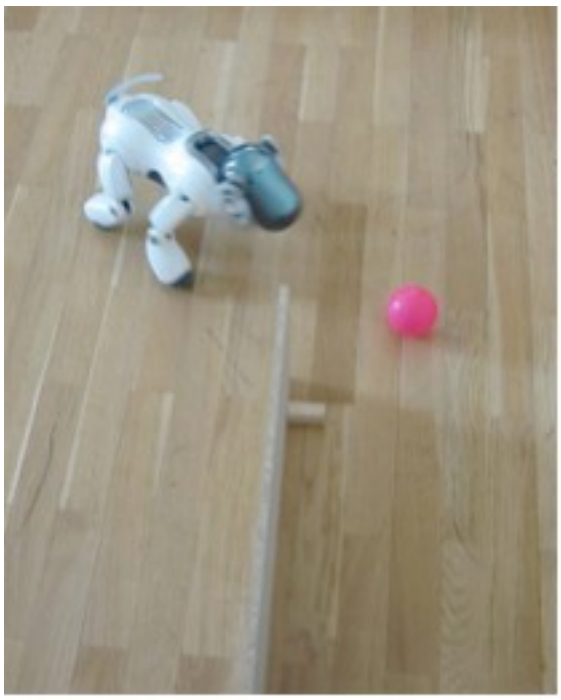


Figure 46: AIBO watching the IBall disappear behind a wall.

6.1.4 Scenario Level three: Hunter-Prey Scenario

The purpose of this task is to develop more complex behaviours, and extend the anticipatory capabilities of the AIBO. This is by far the most complex level of the scenario, whereas the complexity can be varied easily. In the beginning, the AIBO will observe the prey and consequently try to intercept it. The scenario can be extended, wherefore the robotic agent might then observe the prey and learn typical "hiding places", later on moving to anticipated "hiding places" of the prey when trying to catch it.

Objects: In the beginning only KURT 3D, later several walls and obstacles, suitable as hiding places.

[3] For more details see D4.1, section AIS – "how planning/anticipation could emerge"

Task: Prove anticipatory capabilities; extend acquired concepts and generalisations.
Results: To develop higher behaviours, e.g. finding and retrieving an prey/object, and explicit planning for action decisions.

Complexity:
This task involves approaching the following problems as:

- Object recognition
- Motion tracking
- Selective attention
- Relation extraction and encoding …
- Map building
- Generalisation from previously observed
- The AIBO should be able to (at least implicitly) categorise observed objects, regardless if only parts are observed (recognition of semi-visible objects behind other objects, e.g. KURT 3D hiding behind an obstacle, being partially visible)

Figure 47: AIBO watching KURT 3D

Prediction/Anticipation: Predicting and anticipating the prey's actions (see above).

6.2 OFAI Environment

6.2.1 Simulation versus Reality

The issue whether it is more viable to use a simulated autonomous mobile agent or to make the step into the real world is not really new and broadly discussed. Both approaches have their advantages and naturally also their drawbacks. Brooks pointed out that "*there is a vast difference (which is not appreciated by people who have not used real robots) between simulated robots and physical robots and their dynamics of interaction with the environment" (Brooks 1991)*. Most of today's robots work in modelled worlds, instead of the real worlds, what is feasible because it allows focusing on specific issues, instead of dealing with dozens of unsolved problems. As described above, the OFAI test bed is a simplified model world, adjusted to the ecological niche of the robots.

The most obvious choice would be to combine the advantages of simulated robots with real embodied agents by using the simulated robot first and then run the control programme on the physical agent. Even though this sounds feasible, there are still disadvantages (Brooks 1991):

- *"Without regular validation on real robots there is a great danger that much effort will go into solving problems that simply do not come up in the real world with a physical robot."*
- *"There is a real danger (in fact, a near certainty) that programs which work well on simulated robots will completely fail on real robots because of the differences in real world sensing and actuation – it is very hard to simulate the actual dynamics of the real world."*

Simulated agents

The major drawback about interacting with the real world when using reinforcement learning (what is part of the AIS architecture proposed by OFAI) is the large number of runtime trials necessary, and the need for carefully "shaping" the learning task by it into small pieces that the robot is able to learn sequentially. Brooks points out that it seems that for real animals, the vast number of trials necessary is spread over generations, and runtime learning has a more constrained space in which it must search.

Of course simulation is also cheaper and faster than the use of real robots, especially when considering that a simulated robot cannot be damaged or damage its environment and that the simulation can be sped up easily.

Additionally parallel execution of simulation offers another appealing possibility. Different parameter sets can be run simultaneously in several programme tasks, instead of letting one robot solve those problems sequentially.

Drawbacks, when using simulated agents instead of the real world, are pointed out by (Brooks 1991) as follows: First, there is no notion of the uncertainty that the real world presents. Second, there are tendencies to postulate sensors which return perfect information (e.g. Krautmacher and Digler experiment in their simulator using special sensors for identifying life signs of human victims, or sensors detecting food (Krautmacher and Dilger 2004)[4]. No real sensors can perform such complex tasks), and very often the global world view and the robot's sensor view are mixed up.

Additionally using a simulator may lead to problems which do not occur in the real world (Brooks 1992). A simple example is the encounter of two robots in a grid world. In this case the path planner has to solve a stalemate situation. Normally robots do not arrive at the same time at the same position; hence the phenomenon needs no solution. Also when using non-grid worlds, i.e. a two-dimensional Euclidean space, papers have been introduced, presenting elaborate protocols to avoid deadlocks. However in the real world two robots hardly ever run down their corridors perfectly and arrive at identical times, creating such deadlocks.

[4] Please note that Digler and Krautmacher postulated perfect sensors because their primary interest was to verify a new immune system approach for a rescue robot and not having to struggle with issues on sensor level.

Real (robotic) agents
In the real world, the sensors of an embodied agent deliver very uncertain values, even in a static world, since components of the same type are never the same, two motors have different acceleration, two cameras return diverse colour maps, infrared- and ultrasonic sensor values for the same situation differ, etc. (Nolfi *et. al* 1994). Simulations hardly ever take this into account, but however it represents a crucial fact about real embodied agents.

The substantial attribute to be modelled is noise and uncertainty of perception and action. Wheels of the robot may slip, the same driving force may result in various advance speeds on carpet or parquet, and even the direction of carpet fibres may play a significant role. A signal emitted by the ultrasonic sensors is reflected differently from different surfaces reporting divergent distance information. Because of reflections signals may even reach the receiver of another ultrasonic sensor, resulting in an evidently incorrect value. The signal's angle of incidence can also be of relevance.

Even without these difficulties, it has to be mentioned that sensor readings do not represent a direct description of the world. Sensors measure certain quantities or indirect aspects of the world. Sensors do not have any concepts of the world or objects in the world and thus do not identify objects or return information about objects, they do not even separate objects from the background. When dealing with mobile robots, the uncertain motion of the robot complicates dealing with sensory information. A robot operating in the real world needs a complex perceptual system. As argued by Pfeifer and Scheier (2001) and Brooks (1991) it is impossible to treat perception as a black box with a clean interface to the rest of intelligence. This implies that, in order to be able to deal with the real world, it is necessary to verify algorithms on real agents.

Past experience with transporting an AIS approach from simulation to a real mobile robot
This section is a brief excerpt of experiences gathered when transporting a recent artificial immune system from simulated to a real KURT2 robot.

In the beginning an existing artificial immune system approach generated feasible results when controlling a simulated robot agent, but incipiently did not work on the real robot and in the course of time generated strange results. Finally, after several experiments and adaptations, a stable version of the AIS for the real robot was implemented.

When porting the AIS control architecture from simulation to a real robot the sensor issues were most prominent. The cause of these problems was that objects closest to the sensors often were not made of ideal materials, and thus lead to incorrect or no sensor values at all. Certain reflecting surfaces confused the infrared sensors, what took quite some time to figure out, because the robot did not take a turn for several times, even though it was supposed to learn taking turns; however nothing was wrong with the AIS, but the surface of the wooden-boards in the corners reflected the sensors too well, so that the agent did not perceive it.

Another unpredictable and confusing case was the following: Once the robot was supposed to learn basic obstacle avoidance and wall following and could only choose from the actions go forward, turn left and turn right, when it came up with going backward in certain evasive situations. After some time and explanatory attempts, the solution was a defect on the robot: A screw which fastened a mounting hub on the flattened part of the drive shaft of the gear head motor was loose and thus the turning-behaviour was affected. Based on the autonomy of the AIS a learning process occurred, coping with the indeterministic hardware, what naturally made the hardware defect difficult to detect.

Overview and summary – robotic agents vs. simulated agents
The following table summarises and contrasts using robotic agents versus simulated agents.

Criterion	Robotic agents	Simulated agents
PHYSICAL SYSTEM		
Agent	Agent must be physically built and run; great potential for breakdowns, slow, cannot be run in the absence of experimenter	Arbitrary number of copies can be produced; well-suited for systems involving many agents and artificial evolution; functions reliably even in the absence of the experimenter
Physical environment	Given; environment has its own natural dynamics	Everything must be taken into account by programmer; often hard to simulate; realistic simulations computationally expensive
Sensors	Given; no idealization, no "cheating"; often unpredictable effects occur (interference, reflectory properties of surfaces, drastic changes in intensity)	Sensors hard to simulate realistically; idealized sensors common, e.g., distance, object or agent recognition
Motor system	Dynamics given; complex ones hard to build and hard to control; imprecisions	Dynamics hard to simulate realistically
Dynamics in general	Given; exploitation of dynamics necessary and natural	Hard to simulate; often ignored in simulations; dynamics often not exploited

Table 5: Comparison of real and simulated agents (Pfeifer and Scheier 2001)

OFAI is so far only using the real robots (see later in the document), but due to the reasons pointed out above, using a simulator too for testing options and parameters, co-developing the AIS on the simulated and the real robots is currently being considered.

6.2.2 Robots being used

OFAI is primarily using the Sony AIBO robotic platform (***A****rtificial* ***I****ntelligence Ro****bot***).

Figure 48: Sony AIBO ERS-7 robotic platform

Although AIBO was created initially as a robot for home entertainment, it is being used by more and more researchers looking for a low-cost programmable robot platform. AIBO is completely programmable at a variety of different levels. There exists a broad family of different programming kits for AIBO, which are being used from a wide variety of developers indicating that it is well established in academic circles.

Why are we using the AIBO?

The main advantage of working with Sony's AIBO is that it is an accomplished and stable development platform. In addition, it features state of the art hardware and free and downloadable software-programming tools. This allows fully gearing resources and focus on programming higher levels.

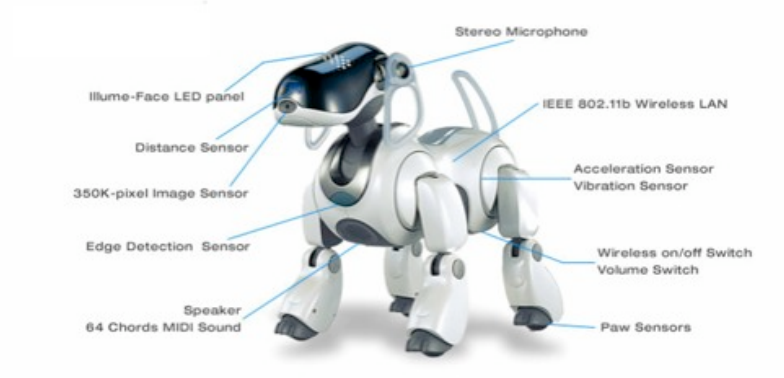


Figure 49: AIBO Features - Front
[Source: Sony AIBO-Europe Homepage]

Figure 50: AIBO Features – Back
[Source: Sony AIBO-Europe Homepage]

Figure 51: Sony AIBO Illume-Face capabilities
[Source: Sony AIBO-Europe Homepage]

The most important specifications summarised:

- Components: Body, Head, Four Legs, Tail, Ears
- CPU: 64bit RISC processor
- Main Storage: 64MB SDRAM
- Program Storage Medium: Sony Memory Stick™ Media for AIBO
- Image Input: CMOS Image Sensor (300K pixel)
- Audio Input: Miniature Stereo Microphones
- Audio Output: Miniature Speaker
- Built-in Sensors: Temperature Sensor, IR Distance Sensor, Acceleration Sensor, Pressure Sensors (Head, Face, Back, Legs and Tail), Vibration Sensor
- Power Consumption: Approx. 9W (Standard Operation in Autonomous Mode)
- Battery Charging Time: Approx. 2 hours
- LCD Display: Time, Date, Volume, Battery Condition
- Operation Temperature 5 - 35 degrees Celsius (41 - 95 F.)
- Operation Humidity: 10 - 80%
- Dimension: 180 x 278 x 319mm (w x h x l)
- Mass: Approx. 1.65Kg (Including Battery and Memory Stick™ Media)

KURT 3D

In the third stage of the OFAI scenario, as final level of the Game Room scenario, a hunter-prey scenario has been suggested. For the realisation of this task, the AIBO robot will take the hunter part and the KURT 3D will be function as prey.

Figure52: KURT 3D and AIBO in the test bed

KURT 3D is an experimental robot platform for sewage inspection, hence the name, which is a German acronym for sewage inspection robot ("*Kanal-Untersuchungs-Roboter-Testplattform*"). Its dimensions are 45 cm (length) x 33 cm (width) x 26 cm (height) and it has an approximate net weight of 10.4 kg.

The robot carries an IBM ThinkPad T42p (1.8 GHz, 512MB RAM, 2kg) and a 3D laser range finder (based on a Sick LMS, +7.0 kg) that increases the height to 51 cm and the weight to totally 22.6 kg. KURT2 operates for about 4 hours with one battery charge (28 NiMH cells, capacity: 4500 mAh). An embedded 16-Bit CMOS microcontroller is used to control the motor and lower sensors (Phytec MiniModul C167, incl. Flash ROM). The maximum controlled velocity the robot can reach using its two 90 Watt Maxon motors (transmission 1:14) is 4.0 m/s (14.4 km/h).

The most important specifications summarised are (according to the KURT 3D homepage[5]):

- 90W Motor
- Power Supply: 38V
- Main Sensor: 3D laser scanner based on a Sick LMS, 181 values in 181 degrees in 13 ms, 24V extra power supply
- Wheel Encoders

[5] http://www.ais.fraunhofer.de/ARC/KURT 3D/

- Maximal possible speed 5.4m/s
- Maximal controlled speed 4.0 m/s. (Controlled speed means that the robot avoids humans and other obstacles)
- Weight: 3D Laser Scanner: 7 kg, Laptop: 4.2 kg, KURT2 Body: 10.4 kg, Cover: 2.8 kg (the cover includes batteries for the 3D scanner)
- Additional sensors: 2 Logitech QuickCam® Pro 4000 VGA-Cameras

7 UW-COGSCI

FINDING AND LOOKING FOR

- ***Finding a specific object (Game Room)***

The purpose of this task is to find a specific object in the environment (e.g. a red cube). The degree of detail in the description must be sufficient to define unambiguously a single object, not a class of similar ones. For example "red cube" is to be used in the case when there is a single red cube, and "big red cube" if there are several red cubes with different sizes and only one of them is big.

- ***Finding members of a class of objects by class description (Game Room)***

The purpose of this task is to find any object matching some general or partial description (for example "find a cube" or "find a red object"). As in the previous case, prediction or anticipation can be based on previous experience, recurring spatial relations, etc.

- ***Looking for an object in the House (House)***

This task is placed in the House environment. Coloured light signals might be positioned above/aside passages between rooms. These lights signal if the passage is open or closed, and might have periodic behaviours. In this task the robot's goal is to find an object that is hidden in one of the rooms in the shortest time or using the shortest way. The level of detail in the object's description may vary. In the case of class-definitions of the target, the purpose of the robot is to find any object that matches the given description.

In some conditions, the target's location is probabilistically biased towards certain locations (e.g. red cubes tend to stand on yellow cubes, although not always, or to stay in some rooms).

UW's aim in this project is to create cognitive, anticipatory systems that can learn how to solve challenging tasks with minimal interference by an external teacher. In fact, UW is interested in having an autonomous system – either in simulation or a real robot (termed robot in either case herein) – learn to interact with an outside world. To achieve this, the learning system is biased towards learning an accurate predictive model of how the environment changes – with and without direct interaction of the robot itself. This task in mind, we describe the scenarios relevant to UW, the intended experiments and required implementations to realize the experiments.

7.1 Monitoring an Interesting Scene

Before starting with actual object or environmental interaction, a preliminary task is to monitor certain scenes with one or more objects in them predicting subsequent sensory input. The scenario is intended to evaluate predictive learning structures that continuously predict subsequent environmental input based on current input and internal state. Such experiments will be carried out with 1-D and 2-D simulations of moving objects. These scenes will be generally small – restricting the information content of the images to enable the system to process the given input. Alternatively to a complete scene monitoring, UW intends to cooperate with IDSIA to evaluate the potentials of

fovea-based vision. Figure 53 shows a scene with several moving objects (indicated by a movement vector). Objects may move with constant velocities and change their direction vectors (lossless) upon impact with another object or the walls.

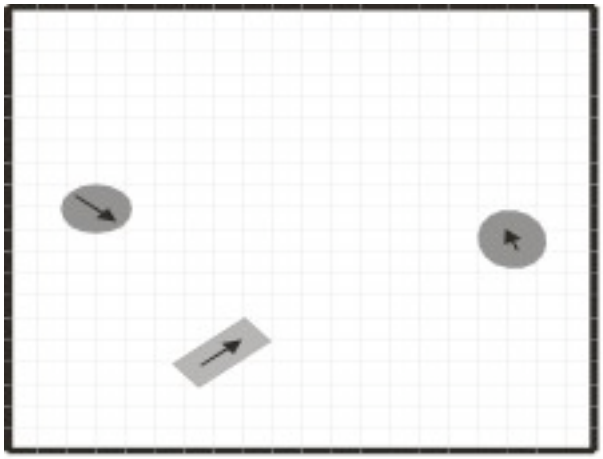


Figure 53 A two-dimensional scene will form the basis for predictive model building experiments.

7.1.1 Representation of the Environment

UW's aim is to represent this environment using the simulated raw camera input. That is, the scene is discretized in small squared areas (as indicated in the figure) where each square is described by a corresponding illumination value. Thus, objects are not directly perceived as objects but rather as darker shades in a light background (or vice versa or just in black and white of with additional color distinctions dependent on the complexity of the task). Considering a grey coded image, a two-dimensional input vector will be provided. The output vector (or the to-be predicted vector) is the input vector at the next state. In this sense, an input vector can be considered as the neural activity on a simulated retina (fovea-based image processing comes at the next stage). Thus, image processing is based from the beginning on (simulated) neural activity based solely on (unprocessed) sensory information.

Alternatively, an abstract representation of the images will be available that denotes exact center coordinates and velocity vectors of the simulated objects. This information may be either used to investigate learning of correlations with visual information and abstract (object-based) coordinate information, or it may serve as a basis of a different kind of predictive learning – based on abstracted, coordinate-based predictive learning. A study comparing learning based on either representation may be of interest to the project.

Finally, each object may be enhanced with different properties such as color, brightness and shape. This information may be available to the learner as well – in any desired code – and may be used to

either encourage the formation of associations between sensory input and abstract object properties, or be the basis for more abstract, symbolic learning investigations.

7.1.2 Learning Object Behavior

It should be clear that either representation (that is, visual input or object coordinates (without velocity vector)) does not provide sufficient information to predict the next state of the objects in question. Thus, the learning system needs to learn an internal representation that predicts continuation of movement once movement is encountered. This movement must be progressed through the predictive model. It is hoped that this leads to an interesting investigation of emergent object-oriented representations of predictive structures. Hierarchical predictive structures are expected to be crucial in the realization of this task.

Initially, it will be most likely sufficient to monitor scenes with one object only. Later on, scenes might be enhanced to monitor various objects with different colors (or brightness / darkness) and shapes. Since any of these properties leads to the need for different prediction types, these property manipulations will serve well to investigate emergent object-oriented representations (in neural network structures or more rule-oriented classifier system structures).

Additionally to property manipulations, the robot may be supplied with both task representations (sensory and abstract) and be encouraged to use the abstract information as context information to improve its predictive sensory system. In this way, associations may emerge that link movement patters from sensory input to corresponding abstract movement patterns, or similarly, movement patterns that are dependent on shapes with abstract shape properties.

7.1.3 Learning Visual Input Change

Apart from monitoring the complete scene, it is also in the interest of UW to investigate the behavior of a system that watches the incomplete scene, or, watches parts of the scene in high detail and other much more fuzzy (fovea-based vision). Similar tasks are imaginable with respect to object recognition. However, the movement of the fovea itself and the expected resulting sensory change are interesting additional challenges.

Movement of the fovea and the focus can both lead to expected and rather regular successive inputs, that is, a change in focus will either decrease or increase the perceptual horizon. Similarly, a shift in focus shifts the image in the other direction. Thus, fovea-related actions can serve as important predictive context information on how the sensory input is expected to change once (and while) an action is executed.

7.2 Finding a Specific Object

Since it is UW's intention to learn emergent object representations, as described above, the first task needs to be evaluated before moving to the task of finding an actual object. Essentially, it is necessary to have evolved a link between plain sensory information and corresponding object properties. Once such a representation is available, the task of finding a specific object can be addressed.

Many ways can enable a system to find an object: In fovea-based vision, it might be sufficient to simply make the system focus on the object in question. Given a manipulandum (such as a robot arm), it would be necessary for the arm to touch the object or even grip the object and move it to a specific position. Given a mobile robot, the robot would need to approach the specific object and touch it touch or grip it in order to trigger a successful "found" event.

7.2.1 Action Control to Find Objects

In any of these cases, it is necessary to have an action control module – or rather have action control integrated into the robot system. Thus, it is necessary to design an action control module that can induce forces (such as directional forces) to influence its own movement (wheels) or the movement of a manipulandum (such as the robot arm).

UW is currently developing a robot arm simulation. Our simple arm model is able to simulate an arm of two or three degrees of freedom being able to manipulate the scene, that is, objects in the scene. The manipulation itself may lead to further capabilities. UW intends to apply this robot arm simulation to our simple 2-D visual simulation in which a robot arm may be added whose end touches the scene – potentially manipulating object in the scene.

Another goal is to enhance the above 2-D moving object simulation by enabling the robot to manipulate one of the objects in the scene inducing force vectors on the object. In this sense, then, the object will be manipulated by the robot directly. The object then represents the robot arm. To find another object in question, it will be necessary to execute certain movements making the arm (that is, the controlled object) move in certain directions. Several simulations are possible either without friction or with friction, where the latter will make the stabilization of the system most likely much easier.

7.2.2 Object Identification

In the simpler scenarios in this section it is assumed that a full description of the type of object to be found is provided. Thus, dependent on the representation the object information might comprise the actual sensory representation of the object. However, an object property representation – that then may trigger corresponding sensory information – might be of stronger interest and poses a bigger challenge to the project.

Certainly, though, the discussed environment enables also the provision of abstract object properties that result in object identification and retrieval. Thus, the environment also supports the detection and retrieval of objects on a much more symbolic level. From the cognitive UW perspective, though, the task will be approached with non-symbolic or only partially symbolic information usage.

7.2.3 Occlusion of Objects

In the so-far discussed 2-D world, no overlaps were allowed. That is, either the object is in the scene and thus visible or it is not present. However, in the fovea-based vision scenario, the object in question may currently not be in the focus of the fovea so that in this case active search actions (such focusing and moving the focus of the fovea) are required.

To actually simulate occlusion, it is necessary to move to a 3-D scenario in which objects can be behind each other, can move past each other etc. It is clear that in this case, first, object behaviors need to be predicted in a more advanced fashion. That is, occlusion needs to be added to the predictive repertoire. Moreover, object permanence is another challenge that needs to be investigated.
To start the investigation of finding objects in a 3-D environment, it might be initially sufficient to start over with the monitoring scene scenario tasks– now in a 3-D context. The scenarios are being developed.

7.3 Finding Members of a Class of Objects

Besides the necessary actions of "finding" a specific object, it is necessary to identify which object needs to be found. In the above discussion a complete object description (dependent on the problem representation used) is already available. However, clearly such a representation is usually not available (at least in such an explicit form) to cognitive systems. Rather, the cognitive system may develop abstract, emergent object properties that then may be linked to motivations to retrieve such objects. Thus, a more challenging task needs to define object properties that are necessarily searched for. Dependent on the complexity of the situation, the object properties may range from anything that moves to actual object properties including color, shape, size etc. In the associative learning architectures, then, the desired properties may be fed into the system from the top (that is, internally) driving the robot to detect the desired properties and then retrieve the localized objects by the necessary actions, as discussed above. In a more advanced stage of the project, then, motivational drives may be included that lead to a drive to interact with certain object, find certain objects, etc. Object occlusions, etc, will make the task even more challenging. Object permanence and an internal world model could enable the robot to form optimal search plans.

7.4 Looking for an Object in a House

The final task in the project will be to combine most of the successfully implemented features and solve problems in a house environment, sketched in Figure 54. The robot's perspective may be restricted to a fovea-like area in which the lighter areas will yield more sensory information than the darker areas. In this way, the robot's perspective (or sensory input) will be localized while the robot's task is to solve global task. The sketched task in the figure would be, for example, to gather as much water and food as possible or keep its internal water and food reservoirs on an acceptable level. Food and water resources may be clustered in certain positions in the house and may be coupled with other resources etc., as desired.

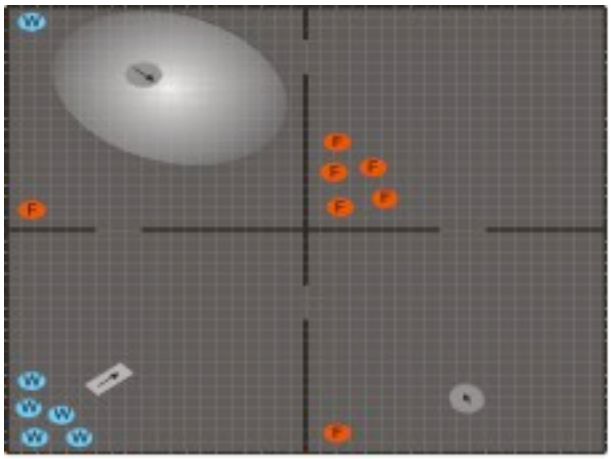


Figure 54 In a house environment, the agent's perspective will be local although it still needs to explore the complete house to gather sufficient water and food resources.

7.4.1 Different Tasks yield Different Challenges

The house environment brilliantly suits for an investigation of deeper inferences and problem solving tasks. As in the room environment, the house environment enables the investigation of goal-directed behavior and search behavior searching for particular objects, object with particular properties or objects that are known to satisfy some internal drives, that is, motivations.

Thus, the robot may be initially tested on retrieving certain objects, such as the red food object depicted in Figure 54. To make the task more challenging, the robot may be asked to switch between retrieving, e.g., red objects and blue objects, alternating between the two. At the last stage of the experiment, motivations may be added to the agent, enabling it to decide on its own which object or type of object to search and retrieve currently.

Many variations are imaginable in the environment including size and color of objects, outline of the rooms and the corridors, location of the objects, etc. Additionally, doors may be endowed with signals or keys to allow passages only in certain cases. Moreover, there are also many agent-related modifications possible such as the importance of filling food and water reservoirs, the effort of moving around, etc. These variations will necessarily be investigated upon and adjusted reasonably. The strategies of the robot will depend on the outline of the rooms, the distribution of the objects, etc. In the interest of the project, it would be interesting to find general distributions and outline properties that require anticipatory behavior to be solved efficiently – or, that at least show that anticipatory behavior is superior compared to high-level planning or simple reactive behavior.

7.4.2 Suitability of Environment and Possible Solution Approaches

The environment suits well to experiment with various retrieval strategies, planning approaches, reinforcement learning approaches, and finally and most importantly, goal-driven, anticipatory approaches. Clearly, it is in the interest of the project to produce effective anticipatory approaches for the problem at hand.

Apart from possibly optimal behavior patterns, particularly initially, exploratory behavior, curiosity, and epistemic actions might be required to learn an effective, abstract, internal representation and use this representation for effective search and retrieval tasks.

Transfer of knowledge may be tested and evaluated, such as special behavior to pass through a door or corridor, or certain retrieval strategies for certain objects, etc. The challenges and possibilities are vast, however, if we are successful in designing a learning system that is able to learn the intended flexible, sensory based, hierarchical, interactive representations, then it is also possible to reason and plan within this representation.

The ultimate goal of this project from UW's perspective is the successful implementation of an agent that is able to learn an effective outline of its environment in an abstracted, hierarchical structure. Moreover, it will continuously try to exploit this structure to improve its own behavior. Motivations will guide this behavior and "pull" the actions of the robot towards those positions or areas in the house that previously led to the successful satisfaction of internal drives.

7.5 Partner Interactions and Conclusions

UW is a cognitive psychology lab that does not have access to actual robots nor does it have the expertise to work with actual robots on its own. Thus, UW is dependent on partner interaction to experiment with the developed algorithms on real robots.

Nonetheless, the proposed simulations are in preparations. A simple simulation of a robot arm is already available. UW is currently working on the proposed simulation environments and specifications will follow.

It is expected that over the next year, the simple one-room simulations and related experiments will lead to first conclusions about the envisioned algorithms and system combinations. These insights will then lay out the further tasks in the project as well as the necessary environmental modifications in simulation and in real robots. It remains to be shown, how far the robot's capability will grow over the remainder of the project. The scenario and the sketched realizations will serve as the guideline to successfully accomplish the challenges ahead.

8 References

Anderson, J. R. & Lebiere, C. *The atomic components of thought*. Mahwah, NJ: Erlbaum, 1998.

Bickhard, M. H. How Does the Environment Affect the Person? In L. T. Winegar, J. Valsiner (Eds.) *Children's Development within Social Contexts: Metatheory and Theory*. Erlbaum (p 63-92). 1992.

Bratman M. E.. Intention, Plans, and Practical Reason. Harvard University Press: Cambridge, MA, 1987.

Brooks, R. A. Artificial Life and real robots. In F. J. Varela and P. Bourgine, (Eds) *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life*. The MIT Press/Bradford Books, Cambridge, MA, 1992.

Butz, M. V. *Anticipatory Learning classifier systems*. Boston, MA. Kluwer Academic Publishers, 2002.

Castelfranchi, C., Modelling social action for AI agents. *Artificial Intelligence*, 103 (1-2), 157-182, 1998.

Drescher, G. *Made-up minds: A constructivist approach to artificial intelligence*. MIT Press, 1991.

Falcone R., Castelfranchi C. Towards a Theory of Delegation for Agent-Based Systems. *Robotics and Autonomous Systems,* 24, 41-157, 1998.

Jackson J. V. Idea for a Mind. *Sigart Newsettler*, 181:23-26, 1987.

Jakobi, N. *Minimal Simulations for Evolutionary Robotics*. PhD thesis, COGS, 1998.

Pezzulo, G., Calvi, G. Dynamic Computation and Context Effects in the Hybrid Architecture AKIRA. In Anind Dey, Boicho Kokinov, David Leake, Roy Turner, *Modeling and Using Context: 5thInternational and Interdisciplinary Conference* CONTEXT 2005. Springer LNAI 3554, 2005a.

Pezzulo, G, Calvi, G. Laila, D. Fuzzy-based Schema Mechanisms in AKIRA. To appear in *IEEE Transactions,* 2005b.

Pezzulo, G., Calvi, G. A BDI with Distributed Knowledge and Control. *ISTC Technical Report*, 2005c.

Krautmacher, M., Dilger, W. AIS Based Robot Navigation in a Rescue Scenario. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) *Artificial Immune Systems. Proceedings of the 3rd International Conference*, ICARIS 2004, Catania, September 2004. Springer Lecture Notes in CS 3239, 106 – 118

LeCun, Y., Cosatto, E., Ben, J. and Muller. U. *Autonomous Off-Road Vehicle Control Using End-to-End Learning.* Technical report Q458, DARPA-IPTO, Arlington, VA, July 30, 2004.

Nolfi, S., Floreano, D., Miglino, O. and Mondada, F. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 190--197. MIT Press/Bradford Books, 1994.

Pfeifer, R. und Scheier, C. Understanding Intelligence. Cambridge, MA: MIT Press, 1999.

Pokahr, A. Braubach, L. Lamersdorf. W. *A Flexible BDI Architecture Supporting Extensibility*, The 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-2005), 2005a.

Pokahr, A. Braubach, L. Lamersdorf. W.. *A Goal Deliberation Strategy for BDI Agent Systems*, Third German conference on Multi-Agent System TEchnologieS (MATES-2005), 2005b.

Rosenbloom, P. S., Laird, J. E. & Newell, A. The Soar Papers: Research on Integrated Intelligence. Volumes 1 and 2. Cambridge, MA: MIT Press, 1992.

Roy, D. Semiotic Schemas: A Framework for Grounding Language in Action and Perception. *Artificial Intelligence*, in press.

Schmidhuber S. and Huber, R. *Learning to generate artificial fovea trajectories for target detection*. International Journal of Neural Systems, 2(1 & 2):135-141, 1991.

Tsai, R. Y. *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, pp. 364-374, 1986.

Verbeek, C., Murr, F., Knoll, A. *Das Robertino-Robotersystem: Ein autonomer mobiler Roboter für Forschung und Lehre.* Technical report I0418, Institut für Informatik, Technische Universität München, Germany, October 2004.

Wolpert, D. M., Kawato, M. (1998) Multiple paired forward and inverse models for motor control. *Neural Networks* 11(7-8):1317-1329.