The Mr.ArmHandOne Project
A mazes roamer robot

M. Cialente, A. Finzi, I. Mentuccia, F. Pirri, M. Pirrone, M. Romano, F. Savelli
ALCOR Laboratory
Dipartimento di Informatica e Sistemistica
UniversitaÁdi Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
{Cialente,Finzi,IvoMen,Pirri,Pirrone,Romano,Savelli}@dis.uniroma1.it

Abstract

We present in this paper the crucial components of "Mr.ArmHandOne" project. We shall discuss topics concerning the robot architecture; the robot building process, including the mechanics and electronics solutions to the robot structure; the description of a dynamic neural network for processing the raw data coming from the sonars; the local map dynamic construction, and the overall structure of the cognitive components. We shall then describe ArmHandOne main task, consisting in £nding and getting some object in a real *non engineered dynamic world*. The test-bed of our research is summarized in the proposed demo: £nding and recognizing objects displaced in an *unknown labyrinthine structure*. The agent is initially positioned in a random place inside a maze, and his goal is to *£nd, grasp and recover a required object, and taking it to an "Exit" position.*

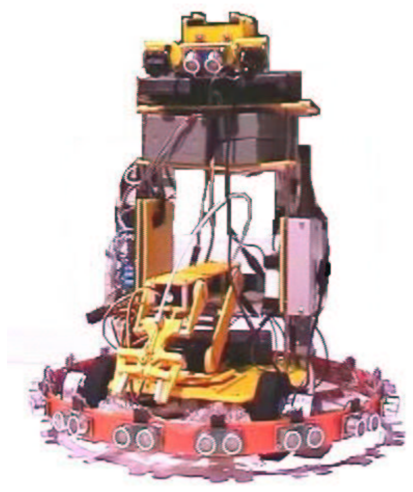

Figure 1: Mr.ArmHandOne

Introduction

In this paper we describe the *cognitive robotics architecture* of Mr.ArmHandOne (Where M and R stand for Mazes Roamer, and the entire nickname should sound like *Mister Armandone*, see Figure 1), and the tasks performed at the AAAI-2002 Robot-Exhibition: exploring a completely unknown maze, picking up objects in the hallways, and £nd the exit (see Figures 2, 5, and 7). Armandone is an autonomous mobile robot roaming non engineered indoor mazes. The real novelty about Armandone consists in his initial knowledge about the world. In fact, since Armandone is prepared to explore a completely unknown environment, his *prior knowledge* is concerned exclusively with physical, structural, topological, and commonsense laws ruling his environment. It follows that he has no map of the domain (the maze) and he doesn't know the real world coordinates of his initial position, but he knows properties of a maze, e.g., that a crossing is a place where two hallways intersect, that the exit is recognizable by a peculiar signal indicating it, that the walls are perpendicular to the ¤oor, that if the map is oriented with the north at the robot left then the south has to be at the robot right, and so on and so forth.

Any fact about the world, like objects positions, distances, shapes, etc., are dynamically acquired while the robot operates to achieve his goal.

Armandone crucial abilities are sensing (via sonars, cameras, and encoders), processing the acquired information for building a model of the environment, and inferring properties about the domain. Here the inference uses the robot's logical attitudes. Armandone's skills, in fact, allow him to £nd a strategy, challenged by imprecise sensors and actuators, for traveling around, and for successfully completing suitable tasks.

The paper is organized as follows. In the next section we shall present the guidelines of the Cognitive Robotics Architecture (CRA). As the architecture is organized in levels which, in turn, are organized in components, there is a section describing each level. Finally, the last sections are dedicated to the description of the exhibition, and the paper concludes with a discussion on the state of the art of Cognitive Robotics.

A Cognitive Robotics Architecture

Our control system is built according to a cognitive multi layered architecture integrating the cognitive and the reactive control. In this paper we stress that a cognitive architecture is essential for the realization of a robot able to perform multiple tasks in the real world, and capable to handle several unforeseen problems, that he would encounter while trying to reach his goals.

We intend the CRA as a formal, structural and computational model specifying the knowledge, the behavior and the functionalities which are necessary to a dynamical system for autonomously interacting with the environment, and with other agents, when required. The CRA that we propose, suitably exhibited by the task performed, is designed on three levels: *structural*, *reactive*, and *cognitive*.

Each level, in turn, is speci£ed by the following three components *representation*, *reasoning*, and *perception*.

The distinction between the structural and the other two levels is straightforward. On the other hand between the reactive and the cognitive level there are crucial differences, that we summarize in the following table.

Cognitive representation:	Reactive representation:
Symbolic Qualitative Discrete	Analytic Quantitative Continuous
Cognitive reasoning:	Reactive reasoning:
Logical Bayesian	Functions de£nitions Arti£cial Neural Networks
Deciding Mission planning Explaining Task planning Progressing Forgetting Updating Revising	Kinematics (body control) Path planning Head-Eyes coordination Motion planning Reaching Manipulating Grasping Holding
Cognitive perception:	Reactive perception:
Focusing Categorizing Interpreting Recognizing	Image acquisition Noise Filtering Segmentation Feature extraction

To better illustrate the above tables, we shall consider Armandone mission: exploring a completely unknown maze, £nding and getting a treasure, and reaching the exit. The mission requires, as a fundamental task, the dynamic construction of a map and a continuous updating of the robot current state.

Let us disregard the way the data coming from sonars and encoders are gathered by the structural level, and consider what happens at the reactive level: a dynamic neural network collects the raw data from the sonars and outputs a probability distribution on the obstacles localization. The map delivered by the ANN is *local* as it concerns the immediate surrounding of the robot. These data are suitably integrated with the data collected by the encoders, in so generating a dynamic global map which is continuously under development and is used for both path and manipulation planning. Motion, path and manipulation planning, in turn, need the kinematics of the manipulator to be computed.

Still, at the reactive level, a parallel visual map is generated by processing the data acquired through the two cameras: the vision system, via the reactive perception, is able to determine whether a hallway is free (no obstacles), whether the obstacle is, or not, the object looked for, whether there is a crossing, the number of intersections at a crossing, and whether the exit has been reached. The reactive level, in other words, suitably process (reactive reasoning) all the sensory information (reactive perception) for the motion control, and it computes the *current state of affair*, that is, the robot position and localization: the distance covered, what he sees (just in terms of localization), where he is supposed to be (at a crossing, in front of a block, in front of an exit, too close to a wall, and so on and so forth).

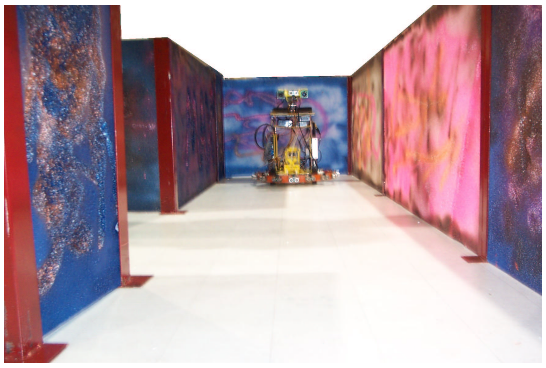

Figure 2: Armandone has been put inside the maze he has to £nd at least one block and bring it to the exit.

Once the state of affair is given by the reactive level, the robot has to *decide* what to do and to *understand* his actual state at a more abstract level, that is, w.r.t. the whole mission, that he has to accomplish. The meaning of *understand*, here, is the following: the robot has to determine whether he knows the exit or not, whether he has already found the treasure or not, whether he has to go back following the same path or not, whether he is inside a loop or not, whether he is lost or not.

All these explanations and decisions require the robot to *reason* about his current state, attributing suitable probabilities to the truthfulness of what he has just gathered. In other words, the robot needs to do theorem proving and Bayesian reasoning. Reasoning, at the cognitive level (i.e. logical and Bayesian), allows the robot to *prove*, given his a priori knowledge and what he collected from his *state of affair*, which properties (of the environment) currently hold, and to what extent, and which properties (of the environment) will hold after executing a speci£c action, and to what extent. I.e. he has to prove, plan and decide.

We shall now describe how we have applied these concepts to the construction of Armandone.

The structural level

The structural level concerns the mechanical, hardware and software speci£cation of the machine structural functioning.

For example the choice of the arm degrees of freedom, the kind of gripper or end effector, the choice of the wheels, breaks etc. for the mobile platform, the kind of cameras and sensors and so on and so forth. It concerns also the hardware and software architecture, although the choices, in the speci£c case, were mainly driven by the limited potentialities of Armandone. On this basis the three components of the structural level can be de£ned as follows:

- *Description*. It deals with the mechanical parts of the robot, so at this point we £nd speci£cation of joints, servos, sensors, and their mutual interactions.
- *Reasoning*. It deals with the electronic components, the methodologies under which they are projected and the robot hardware and software architecture.
- *Perception*. It deals with the sensors control.

The mechanical structure

Armandone body is based on an electro-mechanic structure commercialized by *Lynxmotion inc.*

The carcass is essentially composed by an anthropomorphic arm installed upon a mobile base. Mobility is ensured by two motorized wheels equipped with modi£ed *Hitec HS-422* servos and a passive caster, allowing the robot to perform a differential drive. The robotic arm is composed of three links, and four revolute joints (4 DOF). The end-effector is constituted by a small gripper; all the arm servos are Hitec HS-422 stepper motors.

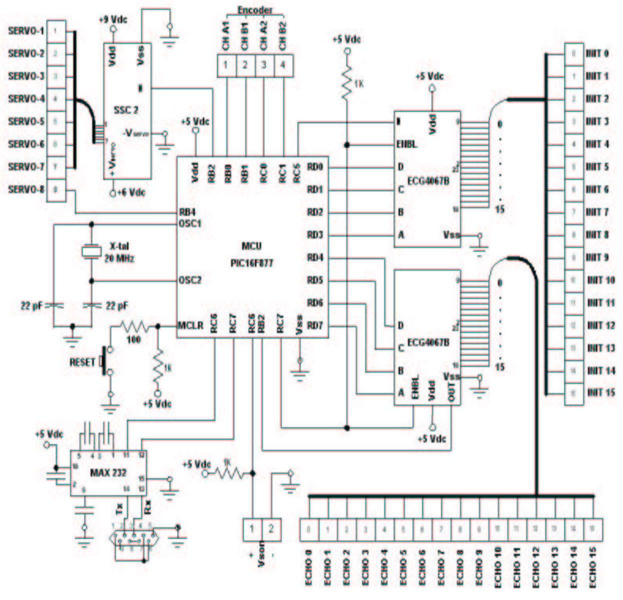

Figure 3: Mr.ArmHandOne embedded system electric schema

Hardware and telemetry

The automaton is endowed with many sensors, and with them the robot perceives the world (exteroceptive sensors), and knows its internal state (proprioceptive sensors).

The exteroceptive sensors are:

1. Two cameras mounted on a pan-tilt head-like support installed upon the base.
2. Seventeen sonars. One is used as a third eye between the cameras, and the other sixteen are distributed along a ring all around the robot, see Figure 1.

The Proprioceptive sensors are:

1. The encoders (see Figure 4), providing high precision information on wheels position w.r.t. the wheels axle, by the way they are engaged to observe movements of the entire robotic structure w.r.t. the surrounding (odometry). Encoders are not skillful, indeed they can't detect longitudinal slipping, neither lateral skidding with the ground, forcing us to use quite carefully metrical data.

Figure 4: Encoder

The other electronic components are:

- An *Electronic board* (see Figure 3) carrying the Central Processing Unit (CPU) with RISC architecture corresponding to the *PIC 16F877*. The CPU collects data from sensors communicating, via a radio modem link, with a remote PC.
- A *Serial Servo Controller Mini SSC II* controlling up to sixteen pulse-proportional motors with a command transmission speed between 2400 and 9600 baud.
- *Two receiver-transmitter*, one for each TV-camera, to acquire and transmit in real time, to the remote PC, a complete binocular vision.

The Reactive level

The reactive level concerns basic and primitive behaviors, i.e. non further decomposable, besides perception and investigation of the nearest space.

Inverse kinematics, for instance, resides in this level. So at this stage robot navigation is adapted to sensed surroundings, and arm joints angle are calculated according to the end-effector position.

Hence the three constituent sub-components are:

- *Description*. It concerns the metric representation of the work-space, local map construction, as well as the representation of the robot con£gurations and details regarding robot dimensions.

- *Reasoning.* It manages obstacles reaction, and it solves the manipulator direct and inverse dynamic and kinematic problems.
- *Perception.* It concerns sensors querying and data processing, in particular metric map learning, features recognition and object focusing.

Navigation: moving around

Armandone rambles around via the two independent motorized wheels and a caster, performing safe tours using his senses: vision, sonars and encoders. Combining motion and sensors abilities we have developed some primitive navigation behaviors:

move(. . .) The robot goes straight ahead until it reaches a given distance, or until it senses an obstruction along the path. The route length is obtained querying the encoders mounted on the wheels, and impediments are felt with frontal sonars.

moveCheck(. . .) This function is almost like the previous *move*, but it adds an important element, in fact *moveCheck* inspects incessantly for an important space characteristic of our system, *lateral discontinuities*, so when the robot lateral sonars perceive an evident irregularity the wheels stop.

turn(. . .) It performs the *turn on a dime* action thanks to the differential drive skill. Particularly useful when the automaton is in a dead-end way.

align(. . .) Performing the *align* behavior Armandone manages the catastrophic orientation errors. The procedure works in two steps: £nding a wall, avoiding corners, and turning the mechanical structure in order to have the robot parallel or perpendicular to the found wall, with an end result to correct the automaton direction.

head(. . .) It regulates the head pan and tilt.

The two cameras are essential for navigation. In fact, *edge detection* recognizes hallways, and discriminate between obstacles and walls. Image elaboration, together with blob analysis, guides also the robot during the last centimeters on the way to pick up an object, and employing the *pattern matching* strategy. Cameras can £nd out useful sign for roaming. Throughout navigation the robot records and works on a metrical local map. This map is output by an arti£cial neural network. Subsequently the automaton merge all his local maps, with the help of local odometric information, to progressively build a global metric map of the work space.

The Local Map: sonar Data on the spot

To build a local metric map of the agent surroundings we use data from the 16 sonars all around the robot. These low-level sensory data are interpreted in a consistent way adopting a neural network, and following the approach proposed by Thrun (Thrun 1998). This methodology applies a dynamic arti£cial neural network to interpreting sonars sensory data: when trained, *the dynamic ANN agrees to characterize every space region near the robot with a probability value representing region occupation degree.*

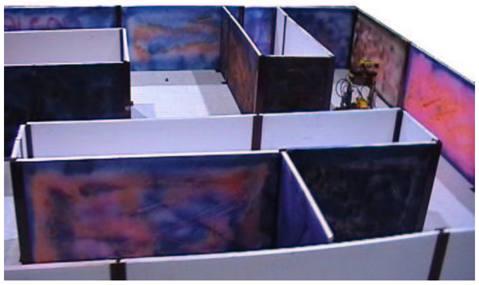

Figure 5: Armandone is looking for the small block, on the left of the picture.

Inputs to the network consist of:

- Two values encoding the position $\langle x, y\rangle$ of the inspected spot in polar coordinates, relative to the robot: the distance from the robot center and the angle from the £rst of the used sonars;
- The three sensor measurements coming from the sonars closest to $\langle x, y\rangle$.

The network can be built during the learning phase and then reduced later (pruning step) until an optimum solution is gained.

To decide if the net needs to grow, we analyze the net error state through a temporal window. If this error ceases to descend quickly enough during the width of this history a new node is added to the hidden layer.

We use two methodologies to create a new node. In the £rst (Wynne-Jones & Mike 1993), the algorithm calculates the relevance of every hidden node, and chooses to split the node with the biggest relevance value. The second one creates a new node if the relevance of the node to be split is less than a threshold, established by experimentation. The algorithm creates a new node in a new hidden layer or a new node in the hidden layer, that has the less number of hidden nodes, and calculates the weights of the new node randomly.

If over£tting occurs, that is to say net error goes under a tolerance value, the net needs to prune. So, considering the same relevance function used for splitting, the node with the least relevance value is found and eliminated.

After a pruning the net will be trained again and, if in the new net the error is bigger than a maximum tolerance value, the algorithm will restore the old net structure.

The network designed in this way performs very well w.r.t. over£tting.

Pick an object: Kinematics

When an interesting object has been individuated and recognized, the robot has to pick it up. To perform the collecting task Armandone is endowed with an anthropomorphic four degree of freedom arm. We have four revolute joints: base, shoulder, elbow and wrist. We achieve the *pick and place* action conceptually dividing the arm into two parts, a rotating base and a three degree of freedom manipulator.

The £rst step consists in positioning the block, that has to be picked up, in the planar work space of the 3R manipulator; this position is reached at £rst with the wheeled base, and then the approach is re£ned, working with the rotating arm base.

The second step requires the computation of the inverse kinematics of the arm. In other words, we have to £nd a suitable arrangement of each single joint, that £ts with the given end-effector position. The manipulation problem requires a representation of the position and orientation of each single link of the arm, and how these links reciprocally interact. Since a stiff body in the space is fully described by the position vector of one of its points, and the orientation of a triple located on it (local-triple), w.r.t. a world-£xed Cartesian triple (world-triple), we start assigning local triples to our arm according to the Denavit-Hartenberg convention (Denavit & Hartenberg 1955). On the basis of geometrical and algebraic considerations, once the links length are assigned, likewise the desired end-effector position and orientation w.r.t. to the base triple, it is easy to determinate the angles for all the three revolute joints. The end-effector orientation is not imposed by the *pick and place* task. According to the result of several tests we have decided to keep the grabber as parallel to the ground as the robot structural limits allow.

Vision processing: identifying crossings, the exit and the blocks

We have already described the head and eyes control, the two functions pan and tilt are, in fact, used to suitably move around the head in order to acquire those visual data which are necessary to perform the three basic recognition tasks: *crossing and corridor detection*, *treasure detection* (the block) and *EXIT detection*.

The Matrox Imaging Library (MIL) is used in the early stages of recognition to acquire raw vision data and to perform some basic operations, such us £ltering and edge detection. After that, a speci£c algorithm for each recognition task is applied, in order to extract relevant information.

For the treasure and EXIT detection, blob analysis and pattern matching, both endowed by MIL, are used. In particular, while the £rst one allows for the identi£cation of speci£c regions in the image, based on geometric properties, the second one allows for the recognition and localization of the required pattern (the EXIT sign) in the image.

A reasoning system is implemented by an Algebra of Figures ($\mathcal{AF}$)(see (Pirri & Romano 2002), and used for hallway and crossing detection. The algebra $\mathcal{AF}$ is based on the notion of *primit* (primitive traits), which is considered to be t he elementary feature of an image, and it can be either a straight line or an arc of ellipse. Primits can be further combined using *connection*, *parallel*, *symmetry* and *orthogonal* operators. Once combined they generate more complex features, i.e. *boundits*, *faces* and *aspects*.

Descriptions of several panel con£gurations are given in $\mathcal{AF}$. These allows the robot to reason about the presence of a straight corridor and left or right recesses. Note that in this domain the reasoning process in simpli£ed because elliptic arcs are not mentioned in descriptions.

The Cognitive level

The cognitive level is concerned with the design and implementation of the ontology, and of the methodologies adopted for reasoning, that is, what the agent knows about the world, and how he uses both his knowledge, and perception, to learn new properties about the world.

The formal representation

The design of the ontology requires to de£ne the minimal set of rules governing the agent domain. Adopting the Occam's razor view, these rules should be primitive. In other words, they should be the axioms from which all the abilities and skills, needed by the agent to accomplish his mission, can be derived. The problem is: *is there a set of primitives general enough to be used in any circumstance, that is, in any domain of application?* Unfortunately, so far we have not been able to devise such a minimal and general set. It follows that we still list down a set of primitives according to a speci£c domain. For example, designing the ontology of an indoor maze, even if it has several common aspects with an outdoor maze, requires a knowledge about relative directions, that for an outdoor maze, in which a GPS or an electronic compass could be used, are not needed. Likewise, in an outdoor maze a knowledge about shadows is needed, in order to determine crossings, while in an indoor maze, in which the light is £xed, it would be redundant. It is easy to see that both for shadows and orientation, one de£nition is a speci£c case of the other, however in general it is simpler to think about the speci£c laws ruling a given domain.

We use the Situation Calculus (McCarthy 1969; Reiter 2001; Pirri & Reiter 1999) as our formal tool to describe the agent's ontology and both Ecl^{i}ps^{e} Prolog (of IC·parc) and Golog (alGOl in LOGic) (Levesque *et al.* 1997), wrapped into C++, to implement the agent's reasoning. This is formally de£ned using a particular (speci£c to the Situation Calculus) form of reasoning, called *regression*, that allows any property to be transformed into a sentence holding in the initial state.

The Situation Calculus is a formal theory of actions that we have suitably extended to handle both stochastic actions and sensing actions, therefore introducing probabilities and another level of reasoning (perceptual reasoning) concerned only with sensory information. The ontology is given through three sets of axioms. The £rst set is de£ned by the set of properties and facts that holds about the initial situation (called S_0). The second set is formed by the *Successor state axioms* (SSA) and the third by the *Action precondition axioms* (APA), one for each action speci£ed in the language. The SSA stipulate the effects, on the domain, of the execution of an action, and the APA stipulate the preconditions for an action that have to be satis£ed for executing it. Other axioms are needed to specify the constraints of the state transition: these constraints are independent of the domain.

Representing, reasoning and executing

Like the other levels of the architecture, also the cognitive level is composed of the following three main parts:

- *Representation*. It involves the ontology, therefore the description of the domain salient aspects. It concerns, in particular the description of objects, properties, dynamic laws ruling state transitions (both successor state axioms and action precondition axioms), constraints (unique name for actions, and fundamental axioms), and other axioms concerning the speci£cation of a stochastic domain and perception.
- *Reasoning*. It is speci£ed using classical theorem proving in £rst order logic, in particular regression is the main formal tool. Because the domain is £nite, all the automatic theorem proving is maintained decidable, furthermore as far as the implementation is concerned, negation by failure is used whenever properties of the domain cannot be proved to hold. Unfortunately negation by failure con¤icts with perception, since new acquired data can crash against data obtained with negation by failure. Observe, however, that the current state, delivered by the reactive level is dominant w.r.t. the properties proved by deduction. In other words if a wall is detected and asserted in the current state – as delivered by the reactive level – and the cognitive level proves that there is no wall (because of negation by failure..) then the initial state is updated with the regression of the current state.
- *Perception*. Is concerned with objects and scenes recognition. Here the formal tools are based on recognition by components (many-dimension primitives). The formalization of perception takes care of internal state transitions induced by sensing actions, and by the effects of reasoning about observations. Armandone lacks a formalization of perception at the cognitive level since his domain is limited to blocks and walls, and such a simple domain is completely managed at the reactive level, as far as visual processing is concerned.

The cognitive level involves also the execution monitoring (high level control), that controls the execution of the tasks, non deterministically chosen by the monitor. A task, in turn, is a sequence of actions sometimes generated by a planner (constructed in Golog), sometimes speci£ed by a Golog program. The execution monitor is the outer control which starts the robot mission, and it is written in Ecl^{i}ps^e Prolog. In the following table we quote Armando's monitor.

```
monitor(St):-
  goal(St), !.
monitor(St):-
  task(Task,St,NewSt),
  executor (Task,St,NewSt, CurrSt),
  checkState(CurrSt),
  monitor(CurrSt).
```

Here the term `goal(St)` speci£es the robot mission starting in the initial state St. In this case, the mission consists of exploring a given maze (observe that the maze can be recon£gured any time Armandone starts a new mission), £nding and picking up one or more treasures and stepping on an *Exit* position, indicated by an exit-sign.

The term `task(Task,St,NewSt)` speci£es the nondeterministic choice of one of the expected tasks: exploring the maze, picking up the block, going to the exit. It is interesting to notice that two distinct possible events can happen while the robot is exploring the maze:

1. The exit is found before the £rst block is found (EB).
2. One or more blocks are found before the exit (BE).

Depending on which of the above events happens a given combination of tasks is effectively executed. The term `executor(Task,St,NewSt, CurrSt)` speci£es the transition of states induced by the task execution and, £nally, the term `checkState(CurrSt)` verify the consistency of the current state, delivered by the reactive level, with the knowledge inferred and stored by the cognitive level.

Maze dif£culty

The problem of how best traversing a maze, and the complexity of £nding a path between two positions, is well known in the literature (see e.g. (Aleliunas *et al.* 1979; Hemmerling 1989; Blum, Raghavan, & Schieber 1991; Blum & Kozen 1978; Rabin 1967), and the history of maze traversal can be found in (S. Bhatt & Tayar 2000)). In particular Rabin (Rabin 1967) has proved that there is no £nite automaton, that can thread all £nite undirected graphs (unless some unbounded information can be stored in its vertices). This impossibility result can be overcome by storing suitable information, via the global and topological map. The problem is, therefore, what kind of information can be stored in the vertices, in order to allow a robot to explore any kind of £nite maze, with any number of cycles.

In the case of a *real-world-maze*, the agent has access to an (approximately correct) local map that is dynamically build. In other words the robot build the graph of the maze while traversing it, and it stores information about positions, length of corridors, number of intersections at a crossing etc. This information, however might be ambiguous. The agent might be unable to disambiguate a location, e.g. when passing through the same crossing. Despite the exploration of a maze can be suitably represented as the traversal of a graph, the dif£culty here, consists in the robot ability of singling out edges from the hallways and vertices from crossings. Each time the robot passes through the same crossing he has to recognize that, indeed, is the same. Furthermore a very short hallway could be mismatched with a crossing, and a very wide crossing might mislead the recognition of the intersections (the out-degree of the vertex associated with the crossing). Therefore the problem of reaching a position in the maze can be divided into two sub-problems: 1) Managing the ambiguity of information collected from sensors; 2) managing the errors that can further blur sensory information (e.g. in a short corridor a wrong wall alignement followed by a wrong twist might induce the robot to sign a position that will be no more recovered). Observe that if all the actions were deterministic (i.e. with the expected effect) and all sensory information perfectly reliable, then the problem could be reduced to that of threading a directed Eulerian graph. Obviously, in a *real-world-maze* actions are nondeterministic and sensory information is unreliable. To face the above problems, we had to introduce a notion of maze *dif£culty* based both on a graph and a geometrical model.

The graph model abstracts away metric information, retaining only topological information of the environment, while the geometric model retains only the metric details. As far as the graph model is concerned, the maze can be assumed to be an indirect connected graph. On the other hand, the geometric model has to do with the degree to which a maze has short and quick twists and turns: an hallway of short length can induce more ambiguity than a longer one.

Let $M(V, E)$ be the graph representation of the maze, where V are the crossings and $E \subseteq V^2$ is the set of all hallways. A cycle $C_{i,k}$ in $M(V, E)$ is a path $v_i, v_{i+1}, \cdots, v_k$ such that $(v_k, v_i) \in E$.

The *topological dif£culty* D of the maze is de£ned as follows:

$$D \ = |E| \times e^{|C_{i,k}|}$$

The *overall dif£culty* D_A of the maze is instead measured by taking into account also the hallways, and their length. Let $d = lnD$ and $length(v_i, v_j)$ the length of the hallway (v_i, v_j), then $D_A = f(d, length(v_i, v_j), |E|)$, i.e. it is a function of the topological dif£culty, and of the length and number of the hallways. This function is de£ned for $length(v_i, v_j) > 0$ and it is decreasing, according to parameters depending on d and $|E|$, till the medium corridor length does not reach an optimal dimension and then it increases again.

block →exit							
$d = \|C_{i,j}\| + ln\|V\|$			$length(v_i, v_j)$				
$\|V\|$	$\|C_{i,j}\|$	d	0.5	1.2	1.6	2.1	2.8
5	0	1.6	0	0	0	0	0
9	1	3.2	0.32	0	0	0	0
15	2	4.7	1.67	1.67	1.32	1.32	1.32
21	2	5.0	3	3	2.67	2.67	2.67
24	3	6.2	5.32	5.32	5	5.32	5
exit→ block							
5	0	1.6	0	0	0	0	0
9	1	3.2	0.32	0.32	0	0	0.32
15	2	4.7	2	2	2	1.67	1.67
21	2	5.0	3.32	3.32	3	3	3
24	3	6.2	5.32	5	5	5	5

The above table shows a benchmark on the percentage of restarting to which Armandone is subject, depending on the dif£culty of the maze. There are, indeed, two benchmarks, depending on whether the block was found before the exit or the other way round. The notion of dif£culty of the maze is thus adopted to show the behavior of the robot in terms of numbers of *restarting*. Where a restarting witnesses the inability of the robot to recognize an already visited crossing. We tested each con£guration at least 3 times, and for each con£guration we report the restarts percentage.

In the next section a more detailed presentation of the way the cognitive level operates on the maze exploration, is given.

The challenge

The demo we have been building up serves to test the cognitive architecture. In fact, it involves an active veri£cation of all the levels of the architecture and their mutual interaction. The demo consists of the following steps. First a maze is built using straight or angular supports that glue together two sheets of polystyrene (see Figures 2, 5 and 7). The maze can be as complicated and intricate as one likes. An exit-sign is attached to a wall and small blocks, of the dimension of a lego-block, are thrown on the hallways. When the scenario is ready, Armandone is put inside the maze at a random position and it is started, by switching on the batteries. Armandone is, now, ready to execute his task. At the very start, he sets his directions as follows, independently of his real position: *the pseudo-north is in front of him and it is set as his pseudo-absolute direction.*

Then he starts exploring the maze to £nd the treasures (the blocks), while looking for the blocks he might encounter the exit, in this case he mentally marks the exit position. Otherwise after having recovered a block he continue exploring until he £nds the exit. If in the meanwhile he £nds other blocks he will pick them up and save them in his basket.

The crucial steps of Armandone's life cycle are:

1. Building a metric-local map (to follow a path) and a topological-symbolic map (to mentally mark the places already visited, and reason about his current achievements).
2. Building the current state of affair, by fusing several data obtained from different processes of computation, and reason about it, to actuate an exploration strategy.
3. Checking the consistency of the state with the inferred knowledge. When consistency is lost (the expected state does not match with the perceived state) then the robot restarts, erasing the current map, and positioning him self in an initial position $(0, 0)$ with the north in front.

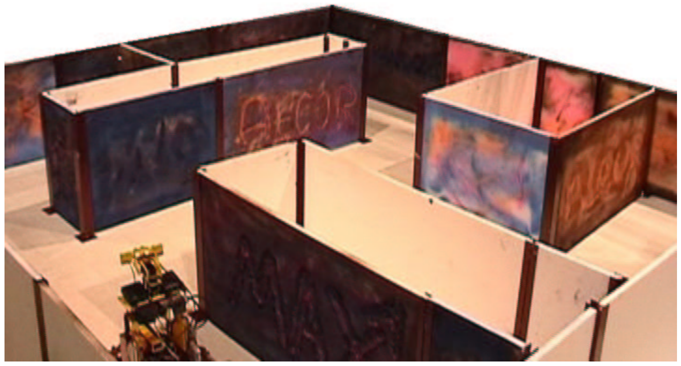

Figure 6: One task has been achieved: picking up at least a block. Now Armandone is looking for the exit.

Tasks and Monitor cycle. At the beginning of every step the robot senses and percepts the environment, builds the local (metric) map, and computes the current state of affairs.

The current state is checked for consistency against the amount of knowledge derivable at the cognitive level. If it is consistent then the current state is used to integrate the current topological map with new data, otherwise the monitor is restarted and all the knowledge inferred so far is erased. The topological map is a set of de£nitions involving either the crossings encountered so far or the pseudo-relative and pseudo-absolute directions of the robot. Remember that at the beginning of the mission, in fact, Armandone sets his pseudo-absolute directions. The pseudo-relative directions are his current left and right. According to the current state a task is chosen. The reactive (and structural) level adapts the selected task according to the environmental conditions (obstacles, discontinuities, crossings ...), and send the commands to the actuators.

Exploration strategy. The maze exploration is performed according to the following strategy, de£ned at the cognitive level:

- Initialize a crossing counter to 1.
- Whenever a crossing is reached, if it is *unknown* (i.e. there is a hallway intersecting the crossing, that has not yet been explored), then *mark the crossing* with the smallest available number n, with the length of the hallway covered so far (from the previous crossing) and mark the number of unexplored hallways intersecting at the crossing.

  If the crossing is *known* (i.e. all the corridors intersecting the crossing have been explored) then go back to the £rst unknown crossing, numbered n-k, $k \geq 1$, delete the numbers marked on all the known crossings encountered in the way back, and suitably update the counter.

  If all the crossings are known then:

  . If the exit is known and a block has been recovered, then stop the exploration and *plan to reach the exit*.

  . If the exit is unknown or no block has been recovered, then an error occurred, restart the monitor according to the following conditions:

    * If a block has been recovered, then restart with the proviso that only the exit has to be found, and erase the current topological map.
    * If no block has been recovered, then restart the monitor just erasing the current topological map.

  At a crossing choose an unknown path according to the following priorities: "go straight", else "go right", else 'go left".

If the treasure has been found the robot must plan to pick it up and put the object in the basket; this task involves the kinematics of the manipulator and manipulation planning, already described in a previous section (see the Reactive Level pg.3).

Planning to reach the exit, on the other hand, is achieved according to the planning strategies, described in (Reiter 2001; Finzi, Pirri, & Reiter 2000) and implemented in Golog.

Discussion

Cognitive Robotics addresses the design of an "embodied arti£cial intelligence" and demands a great effort both in developing new concepts, methodologies, tools and techniques, and in putting them all together. A sheared idea is that high-level control is pervasive: the integration of cognitive skills (reasoning, perception, decisions etc.) in¤uences the whole architecture of the control system. Hence, the embodiment should lead to a more detailed analysis of the interaction between high-level and low-level processes (providing more detailed models for them) in order to make them meet at halfway (e.g. cc-GOLOG language (Grosskreutz & Lakemeyer 2001)). This investigation should yield to a deeper understanding of the relationships among symbolic and numerical approaches.

As a result of this research effort, the high level control system is enhanced with capabilities suitable for managing tasks that are traditionally (Gat 1998; Firby, Propopowicz, & Swain 1998) reactively accomplished. Earliest examples of agent of this kind can be found in (Simmons *et al.* 1997; Burgard *et al.* 1998). (De Giacomo, Reiter, & Soutchanski 1998; Lespérance, Tam, & Jenkin 1999; Finzi & Pirri 2001) emphasize the role of high level monitoring systems, allowing for a strict interaction between deliberation and execution. Here the on-line behavior of the system can be formally veri£ed and adapted to new events in a ¤exible way without affecting the system basic constraint (e.g. in (Ingrand & Py 2002; Williams, Chung, & Gupta 2001)). These approaches aims at providing alternatives to rigid layered architectures where the supervision and execution are managed by separated modules (e.g.(Burgard *et al.* 1998; Gat 1998)). An example of this trend can be observed considering the evolution of the supervision module from RHINO (Burgard *et al.* 1998) to MINERVA (see (Thrun *et al.* 1999)). In the work presented above we discussed an approach to high level control in the context of navigation tasks performed in an unknown environment. High-level control supports navigation and (topological) map building tasks in several ways: checking consistency, driving the exploration, taking decisions in the case of incorrect or suspicious behavior (in (Thrun *et al.* 1999) supervision system plays a similar role) etc. Our exploration strategy can be assimilated to a version of the greedy mapping method (Koenig, Tovey, & Halliburton 2001) managed by the high-level monitor. Furthermore, our architecture enables us to employ the high-level perceptive information, gathered during the exploration, for the topological map building activity. The problem of navigation in an unknown environment can be decomposed into the SLAM (Simultaneously Localization And Mapping problem) and the decision-making about where to go (for example path planning) when the robot needs either to reach a desired location or to explore the environment. Some approaches have been recently proposed in the probabilistic robots programming framework ((Thrun 2000; Thrun *et al.* 2000)). In these works the knowledge about the position and the map is represented by one or more probability distributions, which play the role of beliefs, and are continuously improved through successive Bayesian updates which exploits current data from sensors. Although this approach seems to be very promising, two major drawbacks still remain to be challenged, that are how to scale up ef£ciently to real large and unsafe environments integrating slam and path planning and how to integrate a priori knowledge of the constraints ruling the physical structure of the domain, when this happens to be available. An interesting way for further research could be to investigate to what extent these ideas can be carried in our framework (or vice versa).

Acknowledgments

The Mr.ArmHandOne project is founded by the Italian Space Agency (ASI).

References

Aleliunas, R.; Karp, R.; Lipton, R.; Lovsz, L.; and Rackoff, C. 1979. Random walks, universal traversal sequences, and the time complexity of maze problems. 218–223.

Blum, A., and Kozen, D. 1978. On the power of the corn-pass (or, why mazes are easier to search than graphs. In *Proceedings of the Nineteenth Annual Symposium on Foundations of Computer Science*, 132–142. IEEE Press.

Blum, A.; Raghavan, P.; and Schieber, B. 1991. Navigating in unfamiliar geometric terrain. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, 494–504. ACM Press.

Burgard, W.; Cremers, A.; Fox, D.; Hahnel, D.; Lakemeyer, G.; Schulz, D.; Steiner, W.; and Thrun, S. 1998. The interactive museum tourguide robot.

De Giacomo, G.; Reiter, R.; and Soutchanski, M. 1998. Execution monitoring of high-level robot programs. In *Proceedings of KR'98*, 453–464.

Denavit, J., and Hartenberg, R. 1955. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *ASME J. Applied Mechanics* 22:215–221.

Finzi, A., and Pirri, F. 2001. Combining probability, failures and safety in robot control. In *IJCAI 2001*, 1331–1336.

Finzi, A.; Pirri, F.; and Reiter, R. 2000. Open world planning in the situation calculus. In *Proceedings of AAAI-2000*, 754–760.

Firby, R.; Propopowicz, P.; and Swain, M. 1998. The animate agent architecture. In Kortenkamp, D.; Bonasso, R.; and Murphy, R., eds., *Arti£cial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. AAAI Press/The MIT Press.

Gat, E. 1998. Three layered architectures. In Kortenkamp, D.; Bonasso, R.; and Murphy, R., eds., *Arti£cial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. AAAI Press/The MIT Press.

Grosskreutz, H., and Lakemeyer, G. 2001. On-line execution of cc-golog plans. In *Proceesings of IJCAI 2001*, 12–18.

Hemmerling, A. (1989). Teubner-Texte zur Mathematik, 114. Leipzig: BSB B.G. Teubner Verlagsgesellschaft (ISBN 3-322-00704-9). 215 p.

Ingrand, F., and Py, F. 2002. Online execution control checking for autonomous systems. In *International Conference on Autonomous Systems*.

Koenig, S.; Tovey, C.; and Halliburton, W. 2001. Greedy mapping of terrain. In *proceedings of the International Conference on Robotics and Automation*, 3594–3599.

Lespérance, Y.; Tam, K.; and Jenkin, M. 1999. Reactivity in a logic-based robot programming framework (extended version). In Levesque, H. J., and Fiora Pirri, E., eds., *Logical Foundation for Cognitive Agents: Contributions in Honor of Ray Reiter*. Springer. 190–207.

Levesque, H.; Reiter, R.; Lesperance, Y.; Lin, F.; and Scherl, R. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:59–84.

McCarthy, J. 1969. *Situations, actions and causal laws*. MIT Press, Cambridge, MA.

Pirri, F., and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *ACM* 46(3):325–362.

Pirri, F., and Romano, M. 2002. A situation-bayes view of object recognition based on symgeons.

Rabin, M. 1967. Maze threading automata.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT press.

S. Bhatt, S. Even, D. G., and Tayar, R. 2000. "traversing directed eulerian mazes". In Brandes, U., and (eds), D. W., eds., *In Graph Theoretic Concepts in Computer Science, Proceedings of WG'2000*, 35–46. Lecture Notes in Computer Science No. 1928, Springer.

Simmons, R.; Goodwin, R.; Haigh, K.; Koenig, S.; O'Sallivan, J.; and Veloso, M. 1997. Xavier: Experience with layered robot architecture. *ACM SIGART Bullettin Intelligence* 1–4.

Thrun, S.; Bennewitz, M.; Burgard, W.; Cremers, A. B.; Dellaert, F.; Fox, D.; Hahnel, D.; Rosenberg, C. R.; Roy, N.; Schulte, J.; and Schulz, D. 1999. MINERVA: A tour-guide robot that learns. In *KI - Kunstliche Intelligenz*, 14–26.

Thrun, S.; Beetz, M.; Bennewitz, M.; Burgard, W.; Cremers, A.; Dellaert, F.; Fox, D.; ahnel, D.; Rosenberg, C.; Roy, N.; Schulte, J.; and Schulz, D. 2000. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *Journal of Robotics Research* 19(11):972–999.

Thrun, S. 1998. Learning metric-topological maps for indoor mobile robot navigation. *Arti£cial Intelligence* 99:21–71.

Thrun, S. 2000. Probabilistic algorithms in robotics. *AI Magazine* 21(4):93–109.

Williams, B. C.; Chung, S.; and Gupta, V. 2001. Mode estimation of model-based programs: Monitoring systems with complex behaviour. In *Procedeengs of IJCAI 2001*, 579–585.

Wynne-Jones, and Mike. 1993. Node Splitting: A Constructive Algorithm for Feed-Forward Neural Networks. *Neural Computing & Applications* 1:17–22.