neath" it operate in a reflexive fashion, whereas modules living at higher levels mostly operate by planning. However, the architecture which the system is based upon appears to be flexible with respect to where this boundary is better placed, and it would be very interesting to explore whether – and how – such boundary could be moved any higher – in short, to make "cortical" activities become "spinal" [20].

Acknowledgements

Several people have contributed in many ways to the work described in this paper. Let our grateful thanks go to all the members of the Vision Group at IRST, and in particular to Giuseppe Collini and Dimitri Giordani who have developed and skillfully implemented the hardware and software tools which the system is based upon. Authors feel deeply indebted to Tomaso Poggio, on whose intuitions most of the work is grounded, as well as to Luigi Stringa and Gianni Lazzari for the constant support provided. Many thanks, finally, to the members of the Mechanized Reasoning, Speech Recognition and System Integration groups at IRST, for several helpful discussions and suggestions.

References

[1] T. L. Anderson and M. Donath. Animal Behaviour as a Paradigm for Development Robot Autonomy. *Robotics and Autonomous Systems*, (6):145–168, 1990.

[2] M. Aste, M. Boninsegna, and B. Caprile. A Fast Straight Line Extractor for Vision-Guided Robot Navigation. Technical Report (in preparation), Istituto per la Ricerca Scientifica e Tecnologica, 1994.

[3] R. Brooks. Intelligence without Reason. AI Memo n. 1293, Massachusetts Institute of Technology, Artificial Intelligence Lab., April 1991.

[4] B. Caprile, G. Lazzari, and L. Stringa. Autonomous Navigation and Speech in the Mobile Robot of MAIA. In *OE/Technology '92*, Boston, Massachussets USA, November 15–21, 1992. The International Society for Optical Engeneering.

[5] R. Cattoni and G. Di Caro. ARCA: a Software Architecture to Program Robots. Technical Report (in preparation), Istituto per la Ricerca Scientifica e Tecnologica, 1994.

[6] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333–377, 1987.

[7] J. J. Clark and A. L. Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer Academic Publisher, Norwell - MA, Division of Applied Sciences, Harvard University - MA, 1990.

[8] T. Coianiz and M. Aste. Improving Robot's Indoor Navigation Capabilities by Integrating Visual, Sonar and Odometric Measurements. In Paul S. Schenker, editor, *Sensor Fusion VI*, volume 2059, pages 225–235. SPIE, SPIE - The International Society for Optical Engineering, September 1993.

[9] G. Collini, G. Gabrielli, and G. Avancini. Progetto Piattaforma Mobile. Internal Report 9208-02, Istituto per la Ricerca Scientifica e Tecnologica, August 1992.

[10] J. L. Crowley. Coordination of Action and Perception in a Survellaince Robot. In *IJCAI87*, August 1987.

[11] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[12] F. Giunchiglia, P. Traverso, A. Cimatti, and L. Spalazzi. Programming Planners with Flexible Architectures. Technical Report 9112-19, Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy, 1991. Revised version entitled: "Tactics: extending the notion plan" presented at the ECAI-92 workshop Beyond Sequential Planning (Vienna, August 1992).

[13] L. P. Kaelbling. An Architecture for Intelligent Reactive Systems. In Michael Georgeff and Amy Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Los Altos, California, 1987. Morgan Kaufmann.

[14] R. Kuc and B. Barshan. Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 560–569, 1990.

[15] R. Kuc and M. W. Siegel. Phisically-based Simulation Model for Acoustic Sensor Robot Navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):766–778, November 1987.

[16] Pattie Maes, editor. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* MIT Press, Cambridge, MA, 1990.

[17] N. J. Nilsson. Shakey the Robot. Technical Report 323, SRI International, April 1984.

[18] F. Noreils and R. Chatila. Control of mobile robot actions. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 701–707, Washington, DC, May 1989. IEEE Computer Society Press.

[19] D. Payton, J. K. Rosenblatt, and D. M. Keirsey. Plan Guided Reactions. *IEEE Transactions on Systems, Man and Cybernetics*, 20(6):1370–11382, 1990.

[20] T. Poggio and L. Stringa. A Project for an Intelligent System: Vision and Learning. *International Journal of Quantum Chemistry*, 42:727–739, 1992.

[21] D. A. Pomerlau. *Neural Network Perception for Mobile Robot Guidance.* PhD thesis, School of Computer Science, Carnegie Mellon University, February 1992.

[22] C. E. Thorpe. Mobile Robots. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(3):383–397, 1991.

[23] C. E. Thorpe, M. Hebert, T. Kanade, and S. A. Shafer. Vision and Navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, March 1988.

[24] D. Wilkins. High Level Planning in a Mobile Robot Domain. Technical Report 388, SRI, July 1986.

concerning physical quantities estimated by perceptions could be propagated throughout the Navigation System has been investigated: indeed values of such quantities, which are available only to the Action Manager, may be relevant for either the Navigation Shell (for example, as monitor information) and the motor actions, predicates and perceptions themselves (as optional initialization parameters).

III. 3. Symbolic Layer

Navigation Shell is a "lisp process" running under the Linux environment. Among the Navigation Shell functionalities that we have outlined in Sec. II. 1., all except those reported as point 3 have been fully developed. For these, that is the supervision and interpretation of basic behaviours' execution, much work needs to be done. However, two are the problems we have already addressed: first the interpretation of outcomes of basic behaviours' execution; second, the estimate of robot's location.

- *outcomes of basic behaviours' execution*: any time the execution of a basic behaviour ends, data coming from the Action Manager are interpreted by using a priori information about the expected outcome of the basic behaviour. Such information comes from the map and from a special structure describing certain basic behaviour's functional aspects and aims. This mechanism allows a correct handling of situations in which a basic behaviour's failure - as reported by the Action Manager - should better be interpreted as a success by the Navigation Shell (see for example case reported in Sec. II. 2. and shown in Fig. 3).
- *estimate of robot location*; once a behaviours' outcomes are interpreted, information about robot's location is consequently updated. To this end, not all the piece of information the Navigation Shell can access are to be considered equally reliable, nor it is possible, given their different nature, to make straightforward comparisons among them. For example, although detection of a lateral corridor is a piece of information arguably more reliable than the actual position the odometer returns, no obvious way of merging these data exists. None of the schemes so far devised and tested has proved to perform clearly the best, even though schemes taking advantage of predictions seem to show some advantage on the others.

IV Preliminary Results and Further Extensions

As outlined in the previous sections, the Navigation System at which we have been working is only a part of a bigger network of interacting modules. In this respect, three are the features that most consistently we have tried to pursue: (1) compatibility with the other portions of the MAIA system; (2) openness, that is possibility of extending the functionalities of the Navigation System without any need for radical redesign of the architecture; (3) reliability, that is capacity of surviving critical situations. Preliminary results we have obtained on sequences of randomly generated paths are encouraging, but more extensive testing is needed before any conclusions about the capabilies and weaknesses of the system can objectively be drawn. In the near future, the performance the system is able to exhibit are going to be assessed in the following three scenarios: (1) routine delivery of mail to a specific set of addressees (2) unscheduled, point-to-point delivery of office items (3) telecontrol tasks.

Many are the ways the system we have just described may further develop: the advantages resulting from equipping the robot with more and more sensors (visual, infrared, or tactile, just to cite some example) are interesting. However, using a variety of different sensorial sources does not necessarily solve more problems than it in fact gives rise to. Data of different nature need to be integrated in a unique scheme; their reliability and the different impact that errors may have, need therefore to be appropriately accounted for – something extremely difficult to model, specially when complex interactions among tightly coupled systems are at work. In this sense, the problem slightly differs from a typical data fusion problem [7] since many are the levels at which information coming from different sensors affects the behaviour of the system: at the level of the single component modules, at the level of the Action Manager (for what concerns cooperation among motor actions, predicates and perceptions), or at the level of the Navigation Shell.

On the side of new functionalities to be added to the system, capacity of moving inside spaces less structured than corridors (such as lobbies or office rooms) would of course extend dramatically the practical possibilities of the system. Other functionalities we are presently working at are a *Follow-Me* motor action based on monocular vision, and a set of motor and perceptual primitive for exploration, automatic maps synthesis, and detection and interpretation of written text signs. Although in different ways, all have an impact on the different levels of autonomy that the system is able to exhibit when different working scenarios are considered.

Let us finally mention a somewhat complementary way for pursuing system's improvement that we believe as worth exploring: making basic behaviours evolve towards more complex functional entities. In the present version of the Navigation Nystem, reactivity and planning meet at the level of the Action Manager: modules standing "be-are able to exhibit with different modalities and features. As far as motor actions are concerned, such functionalities are:

1. *Follow-the-Corridor*, to move along corridors;
2. *Move*, to move straight ahead;
3. *Rotate*, to rotate on place.

The abstract predicates consist in:

1. *Covered-Distance-Greater-Than*, signaling when the covered distance becomes greater than a given value;
2. *Rotated-Angle-Greater-Than*, signaling when the angle formed by the robot direction at the activation time with the current direction becomes greater than a given value.
3. *Left(Right)-Corridor-Detected*, signaling the presence of the left (right) lateral corridor;
4. *Bumper-Collision*, signaling when the emergency bumper is pushed;
5. *Unchanged-Location(Direction)*, signaling when at the required time the robot location (direction) doesn't change beyond a given threshold.

The following list concerns the perceptions:

1. *Get-Left(Right)(Frontal)-Wall-Distance*, providing the distance from the left (right, frontal) wall;
2. *Get-Robot-Midline-Angle*, providing the displacement of robot's direction from the corridor's midline;
3. *Get-Covered-Distance* and *Get-Rotated-Angle*, providing respectively the distance and the angle covered by the robot since the activation;
4. *Get-Displacement*, providing the displacement of robot's position from the position assumed at the behaviour's activation (expressed as the triplet $<\Delta x \Delta y \Delta \theta>$).

III. 2. 2. Motor Actions, Predicates and Perceptions: Implementations

Before describing the physical realizations of motor actions, predicates and perceptions, let us make a preliminary remark: our approach is definitely functional, and functional modules therefore provide the basis to the organization of the whole system. In this respect, motor actions, predicates and perceptions can be regarded as the most elementary building-blocks, while no structuring has been specifically defined for the lower-level functionalities which they are made of: the variety of sensors and techniques that can possibly be employed for these functionalities, as well as the difference of complexity and requirements they are asked to satisfy, do not easily allow us to arrange them in a unique yet sufficiently articulated framework.

Among these lower-level functionalities we are currently working at, three are particularly worth mentioning:

- a vision based module aimed at detecting and tracking opening lateral corridors on the image plane, based on intensity and orientation of local edges;
- a module for multi-sensor information fusion which integrates measurements from sonars and images by applying Kalman-like filtering techniques [8];
- a stereo vision based module for obstacle avoidance and landmarks detection.

Looking a little more closely to the physical realizations of motor actions, predicates and perceptions, a set of sonar (and odometry) based implementations of all the abstract functionalities described in Sec. III. 2. 1. has been fully realized and widely tested. Not surprisingly, the biggest portion of work has been required by the *Follow-the-Corridor* module: to move along corridors, avoiding people and small obstacles is a rather complex task to achieve; the motor action has been realized by combining in a subsumption-like way seven simpler motor actions. The sonar based detector of lateral corridors has proved to work properly in the large majority of cases even though some failures still occur in situations of particular crowding which strongly support the additional use of visual information.

At the moment the most interesting vision based realization consists in the perception *Get-Robot-Midline-Angle*: the angle between the robot direction and the corridor midline is measured by computing the position of the vanishing point of the corridor's direction; starting from a gray-level image, intensity and orientation of local edges are computed; significant segments are then extracted [2] and the vanishing point finally located. Particular attention has been paid to robustness and reliability: the perception returns both the computed angle and a numerical estimate of its reliability.

III. 2. 3. Action Manager

Action Manager is a "Lisp process" running under the Linux environment. Albeit rather primitive, it already contains all the functionalities for which it has been designed: the problem of the critical conditions detection has been approached by means of a special set of predicates that are always activated in a background-like modality, even though they do not appear explicitly in the basic behaviour's form; the problem of how the information

In fact, such capabilities are deeply intertwined. All of them make use of a *map* (see Fig. 3) encoding geometric and structural information concerning the environment: geometric information is expressed in terms of spaces that the robot may access; structural information refers to characteristic patterns (e.g. doors, corridors crossings, and so on) which can be used as landmarks.

III The Current System

In this section, the Navigation System is presented according to its latest state of development. The environment which the robot moves in is a portion of the building of our Institute. In particular, all the corridors and open spaces of the first floor are accessible to the robot, and not even the slightest modification of the preexisting environment has been done to facilitate its motion, or the detection of objects and structures.

III. 1. Sensorimotor Layer

The vehicle has been designed and physically assembled at IRST [9]; in this respect, crucial has been the experience made with the fairly popular commercial platform that has served as mobile robot of MAIA during the last three years, and of which the new one represents, from many points of view, a striking improvement.

The external dimensions of the vehicle are 0.5 x 0.5 x 0.9 m and the kinematics is based on two independent wheels plus a pivoting one. The structure of the vehicle – of simple and uncompromising design – consists of three sections: at the bottom, all the electro-mechanical devices and the batteries; in the middle, the control and computing hardware; at the top, sensors and devices for user interface (microphones, sonars, a camera) and the drawers to lodge the objects to be transported.

Computing power is provided by three on board Intel 486 DX2 microprocessors, connected through an ethernet network. The software environment is Unix-like: Linux operating system (non real-time Posix and public domain) on two processors; LynxOS operating system (fully preemptive real-time Posix and proprietary) on the third.

Motors are controlled through a dedicated board, which can return even information on encoders; it also receives stop inputs from both a frontal bumper and an emergency button. The sixteen sonars equipping the robot can detect reflecting surfaces in the range [0.2 m, 4.0 m], and a frontally positioned camera provides standard, 8-bit per pixel, b/w images that are acquired by means of a Matrox IP-8 board. Low-level processing can be performed in order to obtain an image of local edges (represented by two images expressing intensity and orientation of edges respectively) and a list of segments representing the visible borders for incoming images. Some parameters concerning acquisition and processing of sensory data are software-configurable.

Acquisition and early processing of sensory data, data flow among Behaviour Layer modules and actuations of motor commands are embedded in the software architecture *ARCA* [5] – a "C process" runnnig under the LynxOS environment – specifically designed and implemented to make development and testing as simple, standard and modular as possible (see next section).

III. 2. Behaviour Layer

The most delicate problem related to the actual realization of the Behaviour Layer is no new one: making certain conceptually grounded choices underlying the functional architecture coexist with the limitations and the practical issues that any implementation fatally raises. In a case like ours, the problem may indeed become particularly acute, for not only the system is expected to be robust, flexible, real-time and able to support a wide variety of experiments, but it is also foreseen to grow, over the years, bigger and more complex.

In this respect, a great deal of work has been carried out in developing a suitable programming environment, ARCA, in order to support modularity and abstraction from implementation details. When developing a new module (suppose, a motor action), the ARCA environment allows the programmer to completely forget details about the impact that the module being developed is going to have on the computing or memory resources of the system; no deep knowledge is required about the communication protocols that all the modules at the Behaviour Layer and Sensorimotor Layer establish with one another: everything is encapsulated in a software package invisible to the programmer.

At run time, ARCA consists of a multi-thread process in which modules realized by the programmer (motor actions, predicates and perceptions) run as different threads; the input/output between such modules and the physical sensors and actuators is buffered (as well as unbuffered), prioritized and mutually-exclusive. A specialized group of threads manages the interface with Action Manager by executing the (dis)activation instructions, and delivering data and requests between the Action Manager and the modules.

III. 2. 1. Motor Actions, Predicates and Perceptions: Functionalities

Let us recall that each component which a Basic Behaviour is made of can be seen as an element of the set of functionalities that motor actions, predicates and perception make available to the system. Such functionalities represent the abstract capabilities that different physical realizations of motor actions, predicates and perceptions

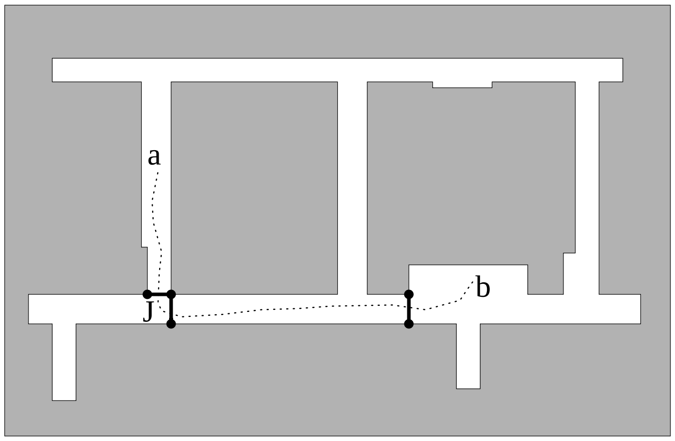

Fig. 3. *The map of the first floor of our institute. Path from* **a** *to* **b** *is expressed in terms of the three "virtual gates" (heavy segments joining opposite sides of the corridors). The dashed line represents a possible realization of a trajectory the robot would actually exhibit.*

inside the building, complying with the constraints expressed in the path. Let us consider the following example: the system receives as input the request of moving from point **a** to point **b**, meanwhile crossing all the virtual gates represented as black segments in Fig. 3. The path is first translated into a sequence of basic behaviours which can informally be expressed as:

1. follow the corridor until a corridor opens on the left;
2. rotate to the left until alignment is reached with the lateral corridor;
3. follow the corridor for 10 meters;
4. move towards point **b** until point **b** is reached.

As long as the robot is actually able to complete successfully all the basic behaviours, the path is traveled in all its length, and the final position is attained. Suppose, now, that as the first virtual door is crossed, some temporary occlusion (people standing nearby, for example) prevents the module committed to the detection of the lateral corridor (a suitable implementation of a *predicate*, in our terminology) to work properly. No corridor is detected on the left, and the robot enters junction **J**, keeping on going straight ahead until it ends up facing the wall. After a few seconds, an appropriate predicate[5] notifies the Action Manager that a stall condition is occurring. The Action Manager quits the motor action, and reports to the Navigation Shell the failure of the basic behaviour. The Navigation Shell, on the basis of the knowledge it holds about the environment, can now perform simple inductive reasonings about whether the goal implicit in the basic behaviour[6] has been attained notwithstanding the reported basic behaviour's failure.

[5]A few predicates able to detect the occurrence of particular conditions (e.g. stall conditions) are always activated together with motor actions, even though they do not appear explicitly in the basic behaviour's form.

II. 3. Remarks

Let us now remark a few points of interest concerning the architecture we have just outlined. Points 1 to 3 refer to general aspects of the system, while point 4 considers specific elements of the architecture.

1. The input-output data structures are of numerical nature, in the form of coordinates within a global reference framework (a map of the environment): this appears to be highly profitable, since it is the lowest level at which the system can be operated consistently abstracting from *how* it works internally;
2. basic behaviours' potentialities may be greater than those presented in the example: the architecture is designed to allow motor actions, predicates, perceptions, basic behaviours and local replanning strategies to grow more and more powerful and efficient;
3. no global information like, for example, a *state* is shared by motor actions, predicates and perceptions across the Behaviour Layer: tasks are performed in a completely autonomous fashion. Information they process is always local and never concerns either global or absolute quantities. This choice is a central one, since it yields a remarkable architectural simplicity and a high degree of independency among motor actions, predicates and perceptions – albeit at the cost of replications of information and processing;
4. in order for the Navigation Shell to fully accomplish the tasks it is designed for, three are the capabilities it needs to be endowed with:
    - reasoning about the implications that activation of motor actions and termination conditions have;
    - processing and maintaining information about the environment;
    - maintaining a consistent representation of the execution.

[6]Notice that the concept of goal associated to a basic behaviour is meaningful only from the Navigation Shell's point of view.

```
(BASIC BEHAVIOUR
    :motor-action Follow-the-Corridor
    :termination
        (OR (Traveled-Distance-Greater-than 10)
             (Corridor-Detected-on-Right-Side))
    :modality Safely
    :monitor
        (ON (Elapsed-Time-Greater-than 5)
              REPORT Covered-Distance))
```

Fig. 2. *Example of a basic behaviour description in Lisp-like notation.*

1. `<motor-action>`, specifying a motor action and – when needed – its parameters;
2. `<termination>`, specifying the termination condition, expressed by combining predicates through logical (Boolean) operators;
3. `<modality>`, conveying how the basic behaviour execution should be carried out (for example "carefully" or "quickly");
4. `<monitor>`, expressing requests for the monitoring of specific events to be performed during execution. For example "report the covered distance after five seconds".

- the *path* is the data structure expressing the location[3] to be reached, and a sequence of constraints (typically, pairs of points that we call *virtual gates*) defining the route;
- the *navigation report* is the data structure that the system returns at the end of the navigation.

From a strictly functional point of view, the whole Navigation System can therefore be thought as a function which:

1. takes a path as input;
2. as a side effect causes the robot to navigate;
3. returns a geometric description of the navigation as it has taken place.

To make the picture complete, two more modules need be introduced: the *Navigation Shell* and the *Action Manager* (see Fig. 1). The *Navigation Shell* is the module that deals with the high-level aspects of navigation. The duties that the Navigation Shell is called to carry out can be schematized as it follows:

1. to determine a "translation" of the input path it receives into a sequence of basic behaviours (the *plan*) whose execution would eventually cause the robot to cross the virtual gates contained in the path, and reach the final location;
2. to deliver, one at time, the basic behaviours to the Action Manager, a module which is primarily concerned with the physical activation and disactivation of motor actions, predicates and perceptions;
3. to supervise basic behaviours' execution, maintaining consistency among information of different nature (functional, geometric, symbolic);
4. to report back to an external agent information about the current state of the system.

The main tasks the *Action Manager* is designed to accomplish are:

1. to determine which combination of "physical processes" (computer programs) implementing motor actions, predicates and perceptions is best suited to realize the basic behaviour being requested;
2. to handle the activation and disactivation of such processes[4]. How to choose among different implementation of the same functionality can be decided on the basis of a variety of elements, such as the actual current conditions of the environment or the sensors being employed (for example, implementations based on ultrasounds sensors would be chosen when the environment is insufficiently illuminated).

As any basic behaviour is completed (no matter whether successfully or not), the Action Manager reports on the execution back to the Navigation Shell. Had something gone wrong during the execution, or some inconsistency had been detected, the Navigation Shell may apply local replanning strategies. When the sequence of basic behaviours expressed by the plan has either been executed completely or it has unrecoverably failed at some point, the Navigation Shell finally issues the navigation report.

II. 2. The System at Work

A typical mission starts with the robot placed in a known location. The robot is then required to reach a given place

[3]The term "location" is used here to mean a triplet (x, y, θ) expressing robot's position and orientation with respect to a given Cartesian frame of reference.

[4]Motor Actions do not end by themselves, but need that an external agent causes them to end.

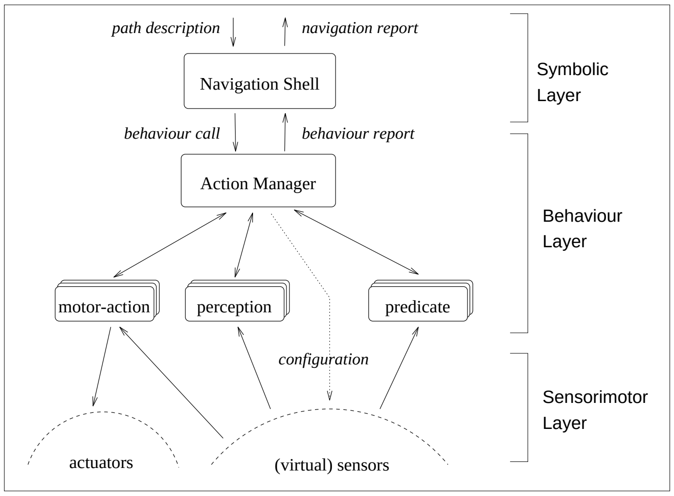

Fig. 1. *The navigation system's architecture.*

system in which a symbolic component controls a purely reactive subsystem. The architecture of the navigation system has therefore been organized in a set of layers, and concepts such that of goal, action, perception and failure have been given suitable – though not always explicit – contextual definitions.

The paper is organized in three main sections. Essential terminology is first introduced in Sec. II, where the architecture of the system is also presented; taking a simple mission as a reference, the way the system works in typical situations is then exemplified. In Sec. III a rather detailed description of the latest realization of the system is contained; its components and the their mutual relationships are presented from either an abstract and practical perspective. In Sec. IV open problems, and directions for future research are finally considered.

II System's Architecture

The Navigation System is structured in three main layers (see Fig. 1): the *Sensorimotor Layer*, which lies at the bottom, and consists of all the modules and devices devoted to the acquisition of data from the sensors and the control of the actuators; at the top of the system, the *Symbolic Layer* deals with the large-scale aspects of navigation, being able to process and deliver to other modules of the MAIA system – namely, the Planner [12] – information about the environment, or the state of the system, in both symbolic and geometric form; in the middle, the *Behaviour Layer* bridges the gap between the first and the second. In particular, it is at this level that modules implementing the abstract activities needed to physically navigate are activated or disactivated, and their functioning coordinated.

II. 1. Terminology

Before entering a more detailed description, a few basic concepts need to be introduced. They are: (1) *motor action*, (2) *predicate*, (3) *perception*, (4) *termination condition*, (5) *basic behaviour*, (6) *path*, (7) *navigation report.*

- a *motor action* is a motor activity aimed at accomplishing a task (such as *Follow-the-Corridor*): this is achieved by activating an appropriate sensors-to-actuators command loop. A physical implementation[2] of a motor action shall therefore map a set of sensorial inputs into a set of commands for the actuators of the robot, in such a way that the resulting motions comply with given prescriptions. In order to be said a motor action, such mapping is further required to be "local" in space and time, i.e. it should not depend on information concerning the large scale structure of the environment in which the robot acts;

- a *predicate* is a perceptual activity aimed at addressing – and, possibly, answering – questions related to events relevant for the navigation. For example: "Is there any corridor opening on your (that is, robot's) right side?";

- a *perception* is a perceptual activity that, albeit not directly related to the control of the navigation, is aimed at providing an estimate of interesting quantities – physical or geometric, such as distances, orientations or intervals of time;

- a *termination condition* is a form made out of a suitable combination of truth values returned by predicates, as, for example: "Have you traveled for more than ten meters *or* do you see a corridor on the right?". Termination conditions typically trigger the disactivation of the motor action being executed, or the activation of other motor actions, predicates or perceptions.

- a *basic behaviour*, is a pairing of a motor action and a termination condition plus a set of optional fields (as shown in the example below). Notice that only the motor action and the termination condition need be specified: the former conveys the motor action that has to be performed, and the latter the events upon which the motor action is to be ended. An example of a basic behaviour's expression (in Lisp-like notation) is shown in Fig. 2. Notice the four elements which it consists of:

[2]The same motor action can be implemented in a number of ways, depending on sensors and algorithms being employed; this is true also for predicates and perceptions.

Bridging the Gap between Planning and Reactivity: a Layered Architecture for Autonomous Indoor Navigation

Roldano Cattoni, Gianni Di Caro, Marco Aste and Bruno Caprile
IRST – Istituto per la Ricerca Scientifica e Tecnologica
Povo, I-38050 Trento, Italy

Abstract— **Research work reported in the present paper concerns the development of a navigation system for a mobile robot designed to accomplish missions inside the building of big organizations, such as hospitals or malls. Useful terminology is first introduced, and a rather detailed presentation of architecture and overall functioning of the system follows thereafter. The most recent realization of the system is then described in some detail, and behaviours and functionalities it can exhibit are reported and discussed. Possible directions for future work are finally outlined, along with open problems that are regarded as the most interesting and challenging for the further development of the system.**

I Introduction

Realization of artificial systems able to move *autonomously*, *purposefully* and *reliably* in fairly unstructured indoor environments is an area of research of relevant technological impact, and may represent a severe testbed for a variety of pure and applied disciplines [22]. It is no surprise, then, if the recent years have witnessed a growing and growing interest for the subject, and if this has in turn yield, beside a number of practical results, a wealth of novel, even strongly unconventional, ideas [16]. Aim of the work our group has been carrying out in the last three years, is twofold: practical, that is the development of the navigation system for the mobile robot of MAIA[1][4]; conceptual, that is the attempt of addressing from an experimental standpoint some among the issues concerning agents-world interaction that one may oftentimes find debated in vague and unnecessarily speculative terms.

In the following, our attention will mostly focus on a system able to receive navigation orders from an external agent (be it a human operator, or another artificial system) in the form of a *path* to be traveled; able, then, to accomplish the order in a reliable and efficient way, and to finally return a suitable description of the navigation, or failure thereof. It is important to remark that the navigation system not only is expected to handle a whole class of events taking place in the external world, but it also needs to interpret appropriate information about the environment, and – when requested – to deliver it to the system that issued the navigation order. Indeed, when compared with the complex network of modules and functionalities forming the MAIA system, that of moving along corridors or hallways becomes anything but a particular one of a much wider class of interactions that we wish a "well educated" robot be able to perform – noteworthy among them all those based on acoustic and visual abilities.

At one end of the navigation system we therefore have abstract, large-scale descriptions of the environment, in which goals, events and functionalities are given symbolic descriptions; at the other, data coming from sensors are transduced into commands for the actuators. In its essence, the problem is how sensing, processing and communication capabilities can be distributed among interacting entities in such a way that transitions from high level goals to elementary commands – and back from elementary actions to high level descriptions – may take place "smoothly" and consistently.

As it is well known, approaches to the action planning problem heavily based on traditional techniques do not generally yield solutions suitable for real-world applications [17, 24], due to either their computational complexity [6] or the inherent unpredictability of the environment. On the other hand, reflex-based approaches that several authors have recently advocated [13, 1, 3] do not generally address the problem of how reflexive behaviours can interface with – or be embedded into – symbolic structures.

The approach we have been following is inspired to earlier works of several authors [23, 18, 19], and consists in a

[1]MAIA is the Italian acronym for *Advanced Model of Artificial Intelligence*, the leading AI project presently developed at IRST.