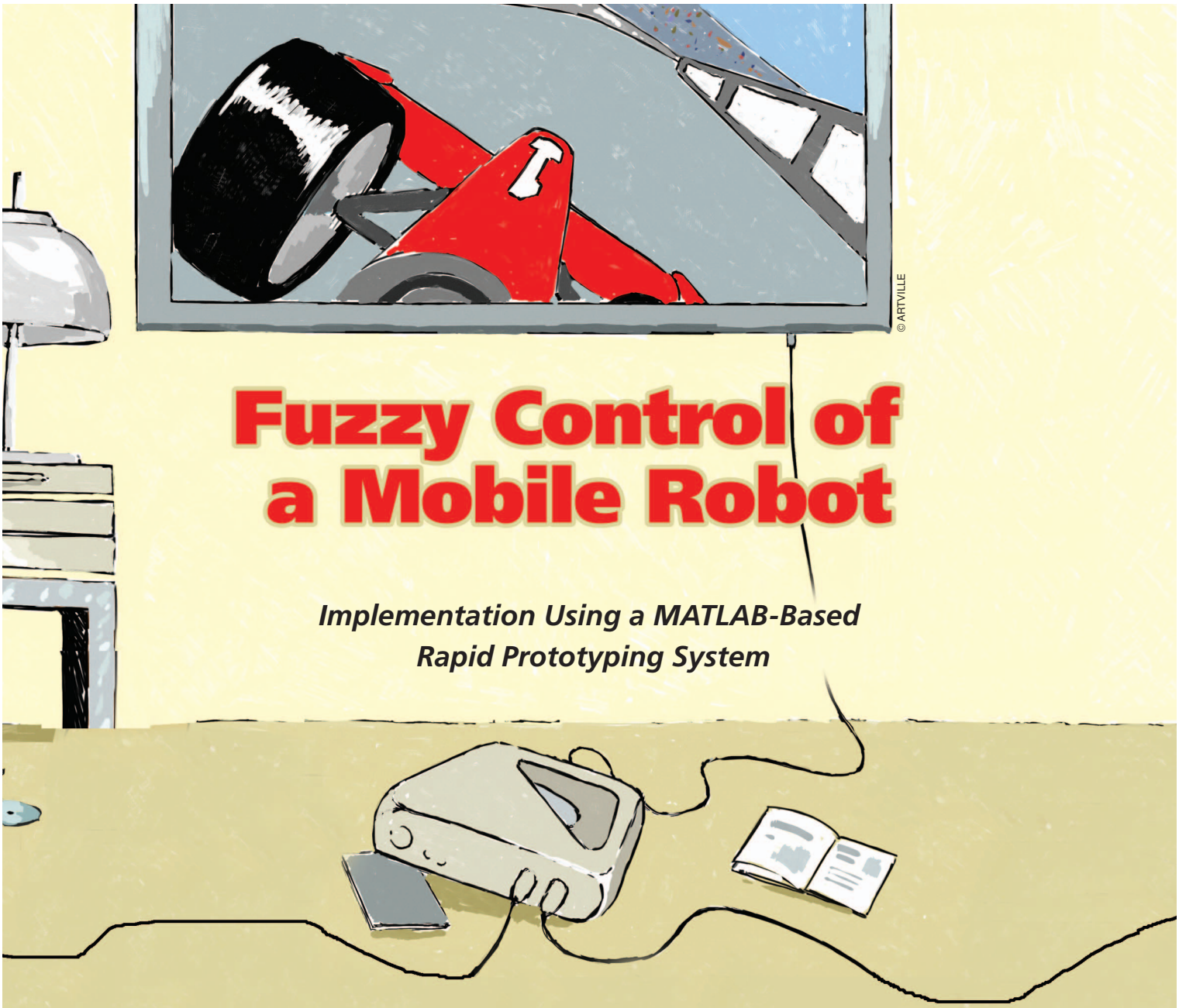

© ARTVILLE

Fuzzy Control of a Mobile Robot

Implementation Using a MATLAB-Based Rapid Prototyping System

The main goal of research on reactive navigation strategies is to allow autonomous units, equipped with relatively low-cost sensors and actuators, to perform complex tasks in uncertain or unknown environments [7]. These technologies have a wide range of potential application fields, which include the exploration of inaccessible or hazardous environments, industrial automation, and also biomedicine. In this research area, the development of the decision and control strategies necessary for autonomous operation plays a central role [4], [7]. Many studies focus on behavior-based approaches, in which the reactivity to unforeseeable circumstances is achieved with computationally simple algorithms that process sensory information in real time by means of high-level inference strategies [7]. In this context, fuzzy logic (FL) is often adopted to overcome the difficulties of modeling the unstructured, dynamically changing environment, which is difficult to express using mathematical equations [4], [8], [9]. Recent examples include the coordination of robot soccer teams [9] and the navigation on rugged terrain [8]. To cope with uncertainties and enhance the robustness of navigation, many researchers have adopted automatic learning techniques, which allow the FLC to exploit sensory data about the explored environment not only for autonomous navigation but also for the adaptation of decision and control algorithms. In particular, computational intelligence methods such as genetic algorithms, reinforcement learning, and neural learning are extremely promising [4].

BY FRANCESCO CUPERTINO, VINCENZO GIORDANO, DAVID NASO, AND LUIGI DELFINE

1070-9932/06/$20.00©2006 IEEE

On the other hand, in contrast to the goal of enhancing robustness and adaptability in real-world environments, it can be noted that a considerable amount of research in the context of fuzzy navigation strategies, and mobile robotics in general, is still based on the design of sophisticated algorithms that are mainly validated with simulations of idealized environments. In particular, these conventional design procedures are based on a sequential schema that includes offline problem description, model development, model-based design, simulation-based debugging and validation, followed by a final implementation on hardware and subsequent trial-and-error refinement. Two fundamental limitations may compromise the effectiveness of such a sequential design procedure. First, it is difficult to give a comprehensive description of unstructured navigation environments and effectively take into account all the details of the unknown scenarios that can have a significant influence on decision and control algorithms. Second, simulations cannot easily take into account the effects of apparently negligible phenomena such as nonlinearities, noise, uncertainties, adverse operating conditions (e.g., poor lighting, defective hardware), and the interaction between concurrent real-time tasks devoted to sensor fusion, noise filtering, decision, and control.

In order to cope with these difficulties, we have developed a flexible and modular hardware platform that allows us to design and validate the fuzzy control algorithms for autonomous navigation directly on hardware benchmarks. The platform addresses the following key issues:

1) It allows us to perform real-time control of commercially available mobile robots, overcoming the inherent limitation of low-cost sensors and actuators
2) It exploits the potential of the widespread and versatile MATLAB/Simulink programming environment [5].
3) It provides a tool for developing user-friendly graphical interfaces for real-time monitoring and parameter tuning in research and education experiments.
4) It combines modular hardware and transparent software architectures, allowing users to simplify configuring, debugging, and refining prototypes.

In this article, the design of a new fuzzy logic-based navigation algorithm for autonomous robots is illustrated that effectively achieves correct environment modeling and noisy and uncertain sensory data processing on low-cost hardware equipment. We have devised a hierarchical control strategy in which three different reactive behaviors [1] are fused in a single control law by means of a fuzzy supervisor guaranteeing robot safety and task accomplishment. Due to the inherent transparency of fuzzy logic, the proposed algorithm is computationally light, easily reconfigurable, and well-performing in a wide range of differing operating conditions and environments.

Platform Architecture

Our platform is based on the widespread mobile robot Khepera developed by the Ecole Polytechnique Fédérale de Lausanne. The Khepera is a differential-drive mobile robot that is particularly useful for rapid initial testing in real-world environments and is currently in use in many universities and research centers to investigate problems ranging from autonomous navigation to biological studies [7]. A Khepera has two wheels, each driven by a dc motor through a 25:1 reduction gear. An incremental encoder on the motor axis gives 24 pulses per revolution, which correspond to 12 pulses/mm of path. The built-in motor controller accepts either position or speed commands. Eight infrared sensors are distributed around the robot for obstacle detection, although they generally provide noisy (and thus sometimes unreliable) signals that are strongly influenced by lighting conditions and by the reflective properties of the obstacles. The available serial link allows for direct access to the readings of the encoders, the proximity sensors, and the position/speed commands.

The RS232 serial wire-link between the Khepera and the terminal unit enables the remote control of all robot functions, thus making it possible to run directly on the remote unit any navigation strategy compatible with the limited speed of serial communication. Alternatively, it is possible to download the control code in the robot's flash memory, even if this solution makes trial-and-error tuning more complicated (a new download is necessary for every modification of the code).

Communication between the remote terminal and the Khepera is carried out by sending and receiving ASCII messages. The terminal plays the role of the master and initiates the communication; the Khepera plays the role of slave and answers only when requested.

Every interaction between the terminal unit and Khepera is composed of:

- a command, beginning with one or two ASCII uppercase letters (representing specific commands for the Khepera) and followed, if necessary, by alphanumerical symbols separated by a comma and terminated by a carriage return (CR) or a line feed (LF) sent by the terminal to the Khepera robot.
- a response, beginning with the same one or two ASCII letters of the command, but this time lowercase and followed, if necessary, by alphanumerical symbols separated by a comma and terminated by a carriage return and a line feed sent by the Khepera to the terminal.

In our platform, we used a dSPACE microcontroller board (DS1104) [3] as the interface device between the PC and the robot. The DS1104 is a general-purpose rapid prototyping control board, fully programmable in a MATLAB/Simulink environment through real-time workshop (RTW) routines. The operating system of the DS1104 includes a set of specific libraries (the mlib/mtrace software) that allow users to set up a real-time communication between the board and the MATLAB routines simultaneously running on the PC. Moreover, the software includes a graphical object-oriented package (the control desk) to develop user-friendly control panels for online monitoring and supervision. Finally, the board features a RS232 serial port directly programmable using the serial communication Simulink blocks. The latter allows users to implement any ASCII-based communication protocol. In our platform, the serial communication for the DS1104 board is developed and configured in Simulink employing the RTI serial interface library. In order to send a command to the robot, all the symbols must be translated

in ASCII 8-b unsigned integers (uint 8). After this, the Simulink serial transmit block is used for sending the obtained bytes to the robot. Similarly, the bytes received from the serial port can be read through the Simulink serial receive block.

To increase the autonomy of the mobile robot and the versatility of the platform, the equipment is completed with a top-view Webcam connected to the PC with a USB interface that provides visual information to the control system. The red, green, and blue (RGB) images from the Webcam are processed directly into MATLAB using color detection codes (image processing toolbox and image acquisition toolbox) [5]. It can be seen that the use of a single programming environment simplifies the processing of visual information and its use in the control strategy. In this way, the robot and target positions are continuously passed to the control algorithm running on the dSpace board through the mlib/mtrace interfaces. On the other hand, since the vision algorithm runs in MATLAB under the Windows operating system, exact real time cannot be guaranteed. However, it is important to note that, considering the limited speed of the Khepera, this limitation is not particularly significant. If the sampling time of the vision system T_v is chosen sufficiently higher than the average image processing time interval, the information received from the camera is always available on time. For our experiments, we chose $T_v = 200$ ms, while the sampling time of the control system was $T_c = 20$ ms.

In Figure 1, a block diagram of the overall test bed architecture is shown. The continuous line represents a physical connection between hardware units while the dotted line represents a data exchange between software modules.

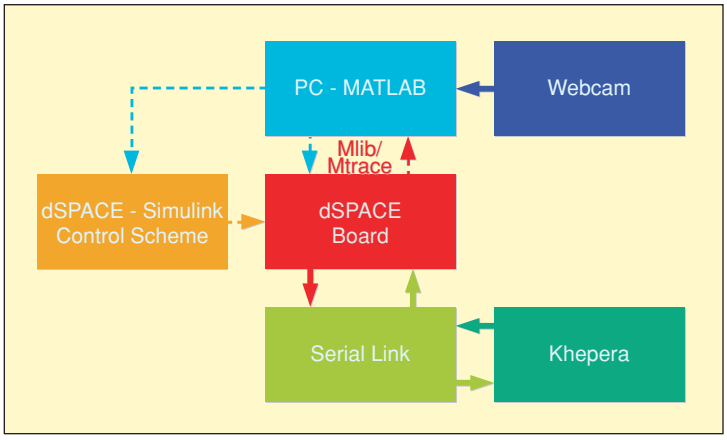

Figure 1. *Overall configuration of the proposed test bed architecture.*

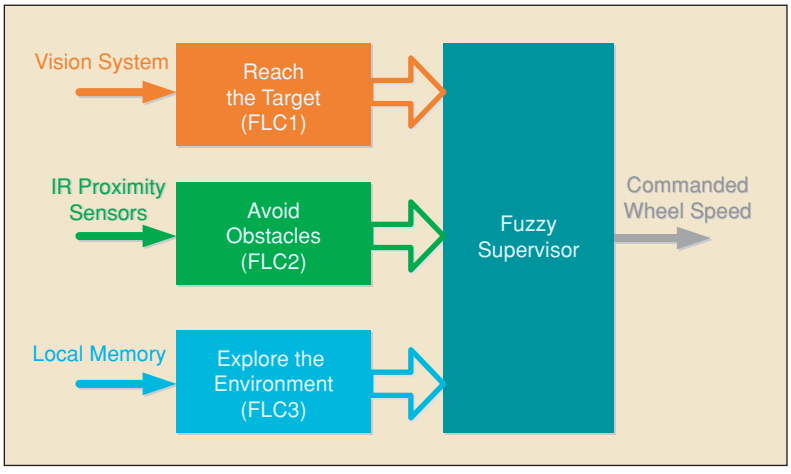

Figure 2. *Behavior-based control scheme.*

Fuzzy Decision and Control Algorithm Design

The Khepera operates in a 1.2×1.2 m^2 arena (viewed entirely by the web cam) with moving obstacles and a target to be reached. The positions of the target and of the obstacles are not known in advance, therefore the navigation algorithm has to implement a reactive paradigm relying only on sensory information. (Interested readers can download videos of various experiments on the proposed platform at the URL: http://www-dee.poliba.it/dee-web/Ricerca/lab-converter/ roboticslab.htm.)

At first, two simple behaviors, namely *reach the target* and *avoid obstacles*, are carried out with two different fuzzy controllers, hereafter called *FLC1* and *FLC2*, respectively. The *reach the target* behavior only depends on artificial vision information and is the primary task for the robot. The *avoid obstacles* behavior uses only infrared (IR) sensor signals. It has the highest priority and takes place only if an obstacle appears on the robot path. Subsequently, a fuzzy supervisor takes charge to combine the reference wheel speeds calculated by each FLC following a priority code. The final commanded wheel speeds are sent to the built-in speed control loop of the Khepera. These controllers are sufficient to guarantee satisfactory navigation performances for the Khepera robot in most of the navigation tasks, except for the risk of getting stuck in box canyons [1]. To avoid this possibility, a further behavior is added to the navigation strategy to prevent the Khepera from visiting the same regions many times. The *explore the environment* behavior makes the robot mark regions already visited and look for unexplored areas. The robot is endowed with a type of spatial local memory, which is used by a further fuzzy controller, henceforth called *FLC3*, to localize and avoid the box canyons. The structure of the whole control scheme is shown in Figure 2.

The modular architecture of our controller has the following main advantages with respect to a monolithic solution:

1) debugging and tuning operations are faster and easier since each behavior is described by few rules and inputs
2) the final structure is more flexible as new simple behaviors easily can be added in order to expand robot skills.

Reach the Target

This behavior reacts to the stimuli of the vision system, providing information about the relative position between robot and target. The behavior ignores the presence and position of obstacles. The Khepera is equipped with two colored markers on top for position and orientation detection, and the target is marked with a red spot. The

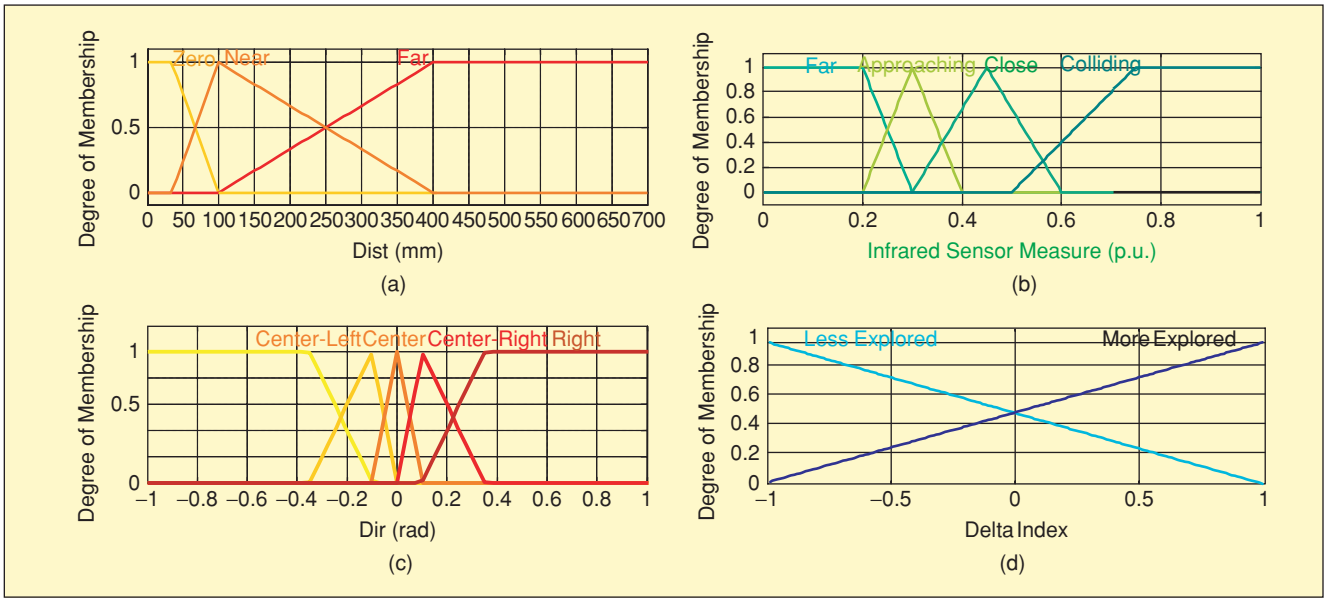

Figure 3. *Input MFs for FLC1 (left-hand), FLC2 (right-hand, upper), and FLC3 (right-hand lower).*

image processing is based on conventional color coding on RGB formats using the MATLAB image processing toolbox. The information captured by the vision system is updated every 200 ms and passed through mlib/mtrace to the FLC1, which is running on the dSpace board. In this behavior, the robot first turns until it is aligned to the target and then moves in a straight line. The information on the distance (DIST) between robot and target and the alignment error (DIR) of the robot is provided by the vision system passed as inputs to FLC1. The linguistic labels for fuzzy sets on the DIST input (expressed in millimeters) are *zero, near* and *far*, whereas DIR input is subdivided into *left, center-left, center, center-right,* and *right* fuzzy sets. The output variables are the speed command signals for the robot wheels and are labeled *negative fast, negative slow, zero, positive slow,* and *positive fast*. For the sake of simplicity, we adopted triangular membership functions (MFs) for the inputs (Figure 3) and five uniformly spaced singletons for the output. In Table 1, the rule base is shown. It can easily be interpreted using AND as a conjunction operator.

Avoid Obstacles

To guarantee a safe navigation of the Khepera, the FLC2 receives the signals coming from eight IR proximity sensors and commands the speed of both wheels so that the robot moves in the opposite direction to close obstacles or goes straight ahead in the case of an open field. The sensors are labelled after their position on the Khepera from S0 to S7 as reported in Figure 4. We used triangular MFs for the eight inputs (see Figure 3) and five uniformly spaced singletons for the output. The labels for each input are *far, approaching, close* and *colliding,* depending on the distance between robot and obstacle. Table 2 shows the rule base of this behavior, where input conditions are combined using fuzzy OR as a conjunction operator (for the sake of simplicity, the rules relative to the back sensors S6 and S7 are not shown). For example, if an obstacle is on the left (the *colliding* membership of the S0 sensor is activated), the right wheel speed is *negative fast*, and the left wheel speed is *zero* to make the robot turn clockwise until the sensor no longer detects the obstacle.

Explore the Environment

Using the information provided by the Webcam, the arena is divided into a point grid corresponding to a matrix, the elements of which record the number of times the corresponding square patch has been visited. Each time the robot enters a square patch, the corresponding matrix element is increased by one. The dimension of each square patch is about 25 mm × 25 mm. An exploration index, representing how often the actual area has been visited by the robot, is then calculated considering the value of the element corresponding to the actual position of the robot. It is then necessary to define the proper correspondence between the elements of the exploration matrix and the arena square patches according to the actual orientation of the robot. If the square patch with coordinate (x,y) is the current location of the robot, the mapping between matrix elements and the northern, southern, eastern, and western square patch relative to the current robot orientation (as shown in Figure 4) is carried out, as shown in Table 3. The information about current robot orientation and position is provided by the Webcam.

Table 1. FLC1 rule table.

INPUTS (logic AND)		OUTPUTS	
Dist	**Dir**	**Right speed**	**Left speed**
Zero	Any	Zero	Zero
Near	Center	Positive slow	Positive slow
Far	Center	Positive fast	Positive fast
Any	Left	Negative fast	Positive fast
Any	Center-left	Negative slow	Positive slow
Any	Right	Positive fast	Negative fast
Any	Center-right	Positive slow	Negative slow

In each sampling time, the differences are calculated between the index relative to the current square patch and the indexes relative to the northern, southern, eastern, and western square patches (henceforth delta north, delta south, delta east, and delta west respectively). It is important to underline here that, in accordance with the reactive approach, no global map of the environment is provided. The robot only uses a local map relative to its current location, using just the exploration indexes of the square patches surrounding its current square patch.

FLC3 receives delta north, delta south, delta east, and delta west as inputs and produces the two wheel speeds as outputs. We used triangular MFs for the four inputs (see Figure 3) and five uniformly spaced singletons for the output. The labels for each input are *less explored* and *more explored*. Clearly, for each input of FLC3, the more negative its value, the less explored the corresponding direction.

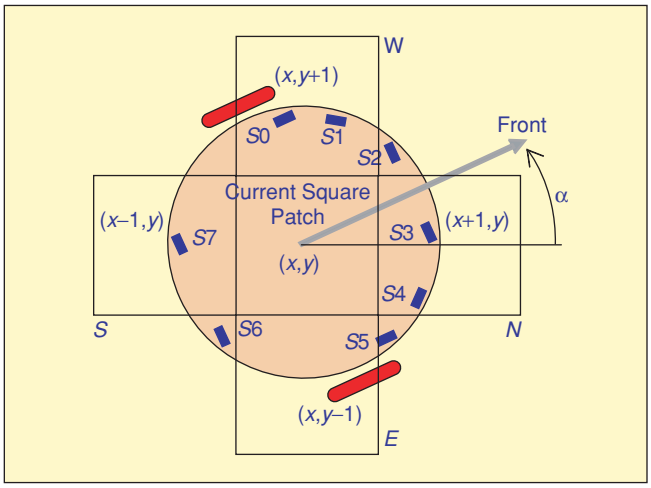

Figure 4. *Position and orientation of the infrared sensors and correspondence between arena square patches and exploration matrix.*

Table 2. FLC2 rule table.

INPUTS (logic OR)						OUTPUTS	
S0	S1	S2	S3	S4	S5	Right speed	Left speed
Coll	Coll	Coll	Any	Any	Any	Negative fast	Aero
Any	Any	Any	Coll	Coll	Coll	Zero	Negative fast
Close	Appr	Any	Any	Any	Any	Negative fast	Positive slow
Any	Any	Any	Any	Appr	Close	Positive slow	Negative fast
Far	Far	Far	Far	Far	Far	Positive fast	Positive fast

Table 3. Mapping between exploration matrix elements and physical square patches with α belonging to [−180, 180].

Orientation	North	South	East	West
$-45° < \alpha < 45°$	(x+1,y)	(x−1,y)	(x,y−1)	(x,y+1)
$45° < \alpha < 135°$	(x,y+1)	(x,y−1)	(x+1,y)	(x−1,y)
$\alpha > 135°$	(x−1,y)	(x+1,y)	(x,y+1)	(x,y−1)
$\alpha < -135°$				
$-135° < \alpha < -45°$	(x,y−1)	(x,y+1)	(x−1,y)	(x+1,y)

The rule table (see Table 4) is based on a potential field approach, as in the case of the *avoid obstacles* behavior, considering more explored areas as higher potential zones or otherwise as obstacles to be avoided. According to this behavior, the robot always moves towards less explored regions and therefore turns right (left) if the region on its left (right) has been already visited, goes straight on if there is a more explored region behind, or turns back on itself if it is facing a more explored region.

Fuzzy Supervisor

In order to safely reach the target, a fuzzy supervisor determines the priority of execution for the three elementary behaviors. This approach is well suited to the task of behavior arbitration because it offers a transparent and extremely effective solution based on high-level linguistic decision rules.

The fuzzy supervisor carries out this task according to the proximity of obstacles and the exploration requirements, as indicated in Table 5. The maximum value of proximity sensors (max_prox), the exploration index relative to the current square patch of the robot (exp_ind), and the commanded speeds of FLC1, FLC2, and FLC3 are the input signals of the supervisor, whereas the outputs are the commanded speeds sent to the Khepera. The labels for exp_ind are *often* and *seldom*, whereas the two labels for max_prox are *close* and *far*. In both cases, the labels are described using triangular MFs.

If the maximum value of proximity sensors is high (the *close* MF of the supervisor is fully activated), an obstacle is dangerously close. Therefore, the robot has to avoid the collision disregarding both the actual location of the target, and the exploration index. In this case, absolute priority is assigned to *avoid obstacles* behavior, while the *reach the target* and the *explore the environment* behaviors are neglected (the outputs of the supervisor coincide with those of FLC2). If the maximum value of proximity sensors is low (the *far* MF of the supervisor is fully activated), then the robot's path is clear. In this case, if the robot is navigating in an already visited area, the target is neglected and the robot starts exploring the environment to avoid box canyons (the outputs of the supervisor coincide with those of FLC3). On the contrary, if the current area is unexplored, the robot points towards the target and only the *reach the target* behavior is

Table 4. FLC3 rule table.

INPUTS (logic AND)				OUTPUTS	
Delta north	Delta south	Delta east	Delta west	R speed	L speed
Less exp	Less exp	Less exp	Less exp	Positive fast	Positive fast
More exp	Any	Any	Any	Positive fast	Negative fast
Any	More exp	Any	Any	Positive fast	Positive fast
Any	Any	More exp	Any	Positive fast	Positive slow
Any	Any	Any	More exp	Positive slow	Positive fast

executed (the outputs of the supervisor coincide with those of FLC1). In every intermediate condition the supervisor will perform a fusion of the three FLCs blending their outputs to achieve a safe navigation toward the target.

Experimental Setup and Discussion

Several navigation experiments were conducted in an arena (Figure 5) with white obstacles (undetectable to the camera) and a red spot representing the target that the Khepera has to reach. Figure 6 depicts the control scheme in which serial communication and fuzzy logic controllers are managed through specific Simulink blocks. The overall tuning of the control scheme can easily be performed by directly changing the parameters in the Simulink scheme, and executing new experiments until the desired robot behavior is obtained.

For brevity, we focus on a sample experiment, chosen to enlighten the particular effectiveness of the hierarchical fuzzy control strategy. Figure 7 shows the performance of the robot during the navigation experiment, in which a box canyon is found in its way. Figure 8(a) illustrates the trend of max_prox and of exp_ind, whereas Figure 8(b) reports the right wheel reference speeds provided by the *avoid obstacles*, the *reach the target*, the *explore the environment* behaviors, and by the fuzzy supervisor. This experiment lasted 80 s, although, for the sake of clarity, in Figure 8 only the time interval ranging from 65–75 s is shown.

In the time interval from 68 s to about 72 s, exp_ind is *often* with the membership degree equal to one. The robot path during this period of time is traced in gray in Figure 7. Since the robot is navigating in an area already visited, the supervisor inhibits the *reach the target* behavior and commands the robot to explore the environment further. Moreover, since in this time interval no obstacles are in the robot path (as can be seen in Figure 8 a max_prox is about 0.2), the *avoid obstacles* behavior is inhibited as well. Thus, the robot can safely move towards less explored areas. As shown in Figure 8(b), in the time interval ranging in 68–72 s, the output of the supervisor coincides with the output of FLC3. At 72 s, since the robot is approaching an obstacle, the supervisor activates the *avoid obstacle* behavior to prevent a collision. From this time on, since the robot has escaped the box canyon, it uses only the *avoid obstacle* and the *reach the target* behaviors to complete the assigned task.

Finally, we want to remark that developing the navigation control code directly on the hardware platform allowed us to

Table 5. Fuzzy supervisor rule table.

INPUTS (logic AND)		OUTPUTS
Max_prox	Exp_Ind	Behavior
Close	Any	Avoid obstacles
Far	Seldom	Reach the target
Far	Often	Explore the environment

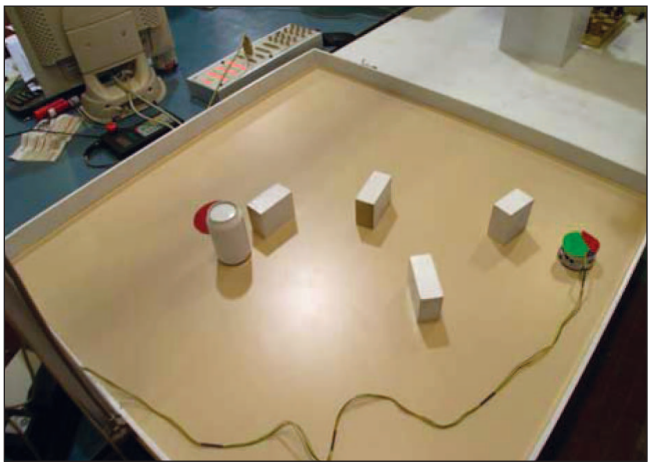

Figure 5. *Panoramic view of the Khepera environment.*

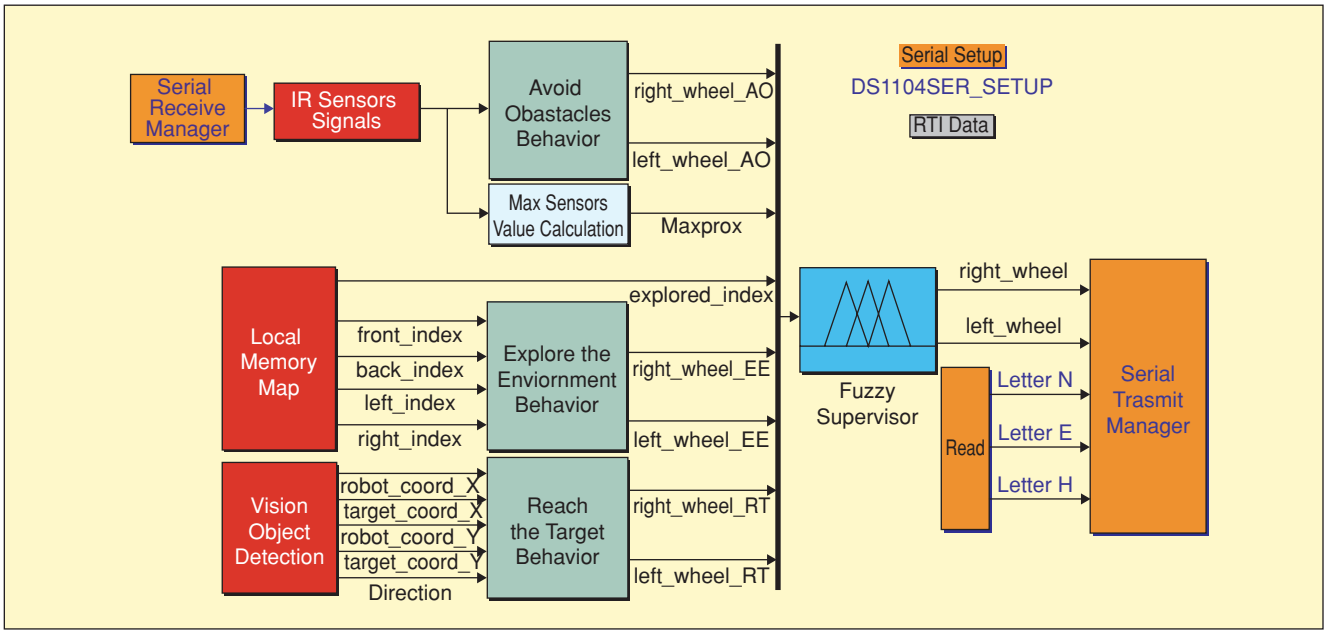

Figure 6. *Real-time control scheme developed in Simulink.*

gain useful insights on the adopted fuzzy design strategies. For instance, in our test, we noted that it was extremely simple to implement each of the three independent behaviors correctly, while the development of the supervisory strategy was more difficult than in preliminary simulations. In particular, the navigation strategy proved to be extremely sensitive to the balance between *avoid obstacle* and *reach the target* behaviors. If the former behavior is emphasized, the robot tends to move away from obstacles, even though they are not directly on its path to the target, and becomes unable to move through narrow passages (exhibiting winding paths). On the contrary, if the *reach the target* behavior is emphasized, the robot tends to cross narrow passages with agility, following straight paths, at the cost of a decreased ability to avoid collisions at the same speed. The platform allowed us to perform without difficulty a sufficient number of experiments necessary to find the appropriate tradeoff between robot speed (avoid obstacle priority) and smoothness of paths traveled (reach the target priority), and, therefore, achieve the presented satisfactory results.

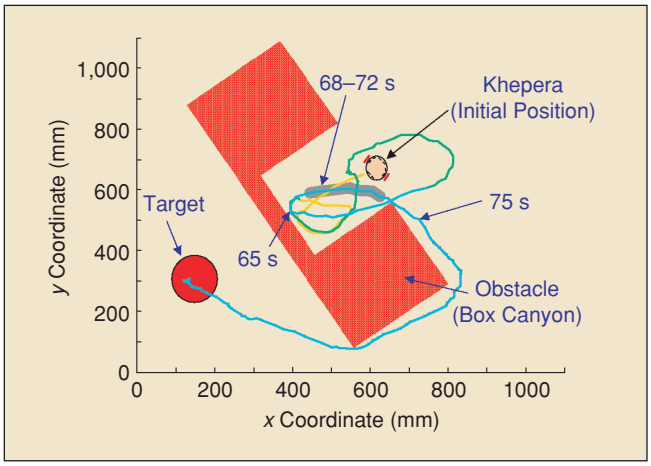

Figure 7. *Measured path of the mobile robot in a real environment.*

Concluding Remarks

The experience of the design of the behavior-based navigation strategy confirmed the remarkable potential of fuzzy logic in the development of transparent and particularly effective decision laws capable of overcoming the inherent limitations of low-cost hardware sensors and actuators. Moreover, the adoption of the rapid prototyping platform is a fundamental means to further simplify and enhance the design process, integrating the various sequential stages (which include time-consuming trial-and-error tuning of the equipment) into a single, straightforward development process. The tuning of the fuzzy supervisor to achieve a satisfactory tradeoff of performance is a typical example of the potential advantages of the integrated design scheme.

We also consider the use of the widespread MATLAB-Simulink software package a key element in freeing the designer from low-level hardware and software issues so as to focus on experimental implementations from a control engineer perspective [2], [6]. Noticeably, the MATLAB-based platform is also an ideal tool for advanced educational activities because it allows trainers to perform straightforward laboratory sessions that bring students as close as possible to real-life challenges, motivating them to develop their studies of theoretical issues related to robotics and control systems. The software/hardware platform can be also profitable for the standardization of laboratory equipment (making it possible to share and reutilize software and hardware easily) as well as for the development of virtual laboratories (remotely controlled experiences for research and education using the Internet can also be set up using the available toolboxes). As regards reusability, it is worth mentioning that the proposed platform was also used to control a modular mobile manipulator composed of a Koala mobile robot (produced by K-Team) and a Katana manipulator (produced by Neuronics). The Koala is a six-wheel mobile robot endowed with 16 IR and four sonar proximity sensors for obstacle detection, and the Katana is a 5-DOF manipulator equipped with IR and pressure sensors on its gripper. In this version of our platform, the serial output of the mobile manipulator was then connected to a Bluetooth

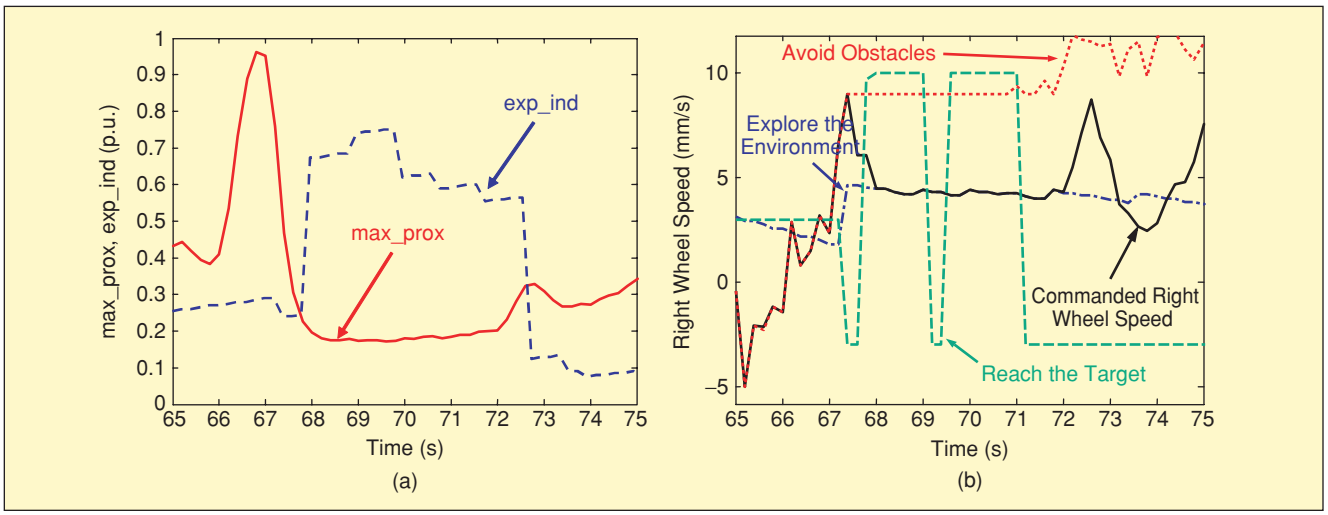

Figure 8. *(a) Trend of max_prox and exp_ind and (b) right-wheel reference speeds provided by the* avoid obstacles, *the* reach the target, *the* explore the environment *behaviors and by the fuzzy supervisor during Experiment 2.*

device for wireless serial communication with the dSpace board. The software modules (Simulink, Control Desk) were obtained by slightly modifying those developed for the Khepera, and also the fine-tuning of the navigation strategy was carried out without difficulty. The platform allowed us to set up the serial communication between the robots and the board, monitoring all the signals coming from Koala-Katana sensors, and simultaneously controlling the robot and the manipulator using the remote unit.

The main limitations of the proposed platform lie in the speed of the serial communication, which is particularly limited in the case of the relatively low-cost Khepera hardware. When additional hardware is installed on such a mobile robot (e.g., equipping it with a gripper and additional sensors), it is necessary to decrease the sampling time of the communication, significantly reducing the maximum speed that guarantees satisfactory navigation performance (note that the K-team radio turret for wireless control is even more limited with respect to communication speed). Moreover, it should be reiterated that since the image processing routines run in a MATLAB environment, it is not possible to guarantee exact real time for the entire navigation strategy. In fact, the image-processing algorithms are the most time-demanding tasks in the present version of the platform and constitute the most limiting constraint for the maximum safe-navigation speed.

Future work on the research project will include the extension of the control approach to communities of mobile robots endowed with a specific set of behaviors including coordination functions.

References

[1] T. Balch and R.C. Arkin, "Avoiding the past: A simple but effective strategy for reactive navigation," in *Proc. IEEE Int. Conf. Robot. Automat.* vol. 3, May 1993, pp. 588–594.

[2] L. Chaves, P. Spiller, A. Scolari, A. Conte, and L. Pereira, "A MATLAB/Simulink-based platform for real-time planning and execution of control techniques applied to mobile robots," in *Proc. IEEE Int. Symp. Comput. Intelli. Robot. Automat.*, 2001, pp. 530–535.

[3] dSpace, GmbH, *ds1104 Microcontroller Board Reference Manual*, [Online]. Available: http://www.dspace.com/ww/en/pub/home/products/hw/singbord/ds1104.cfm

[4] D. Driankov and A. Saffiotti, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, New York: Springer Verlag, 2001.

[5] "MATLAB Toolboxes Reference Guides," The Mathworks, Natick, MA [Online]. Available: http://www.mathworks.com/products/matlab

[6] A. Monti, E. Santi, R. Dougal, and M. Riva, "Rapid prototyping of digital controls for power electronics," *IEEE Trans. Power Electron.*, vol. 18, no. 3, pp. 915–923, May 2003.

[7] R.R. Murphy, *Introduction to AI Robotics*, Cambridge, MA: MIT Press, 2000.

[8] H. Seraji and A. Howard, "Behavior-based robot navigation on challenging terrain: A fuzzy logic approach," *IEEE Trans. Robot. Automat.*, vol. 18, no. 3, pp. 308–321, June 2002.

[9] P. Vadakkepat, O. Miin, X. Peng, and T. Lee, "Fuzzy behavior based control of mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, Aug. 2004, pp. 559–565.

Keywords

Fuzzy logic, control, navigation, MATLAB.

Francesco Cupertino received the Laurea degree and the Ph.D. degree in electrical engineering from the Technical University of Bari, Italy, in 1997 and 2001, respectively. From 1999–2000 he was with PEMC research group, University of Nottingham, United Kingdom. Since July 2002, he has been an assistant professor at Technical University of Bari, Electrical and Electronic Engineering Department. He teaches two courses in electrical drives at Technical University of Bari. His main research interests cover the intelligent motion control and fault diagnosis of electrical machines. He is the author or coauthor of more than 50 scientific papers on these topics.

Vincenzo Giordano received the master's degree with first class honors in electrical engineering in 2001 and the Ph.D. in control engineering in 2005, both from the Politecnico di Bari, Italy. In 2004, he was a visiting Ph.D. student at the Automation and Robotics Research Institute, University of Texas at Arlington, under the supervision of Prof. Frank Lewis. As a Ph.D. student, he was also coresponsible for the organization and start up of the Robotics Lab at the Politecnico di Bari. In 2005, he was a visiting researcher at the Singapore Institute of Manufacturing Technology, Singapore. In 2005–2006 he was a postdoctoral fellow at the Automation and Robotics Research Institute, University of Texas at Arlington, codirecting the activity of the Distributed Intelligence and Autonomy Lab. His research interests focus on intelligent control techniques applied to robotics, industrial automation, and discrete-event systems. He has published seven international journal papers and more than 15 conference papers on these topics. He is presently in the advanced systems engineering branch of Technip, working on the design of control systems for petrochemical plants.

David Naso received the Laurea degree (with honors) in electronic engineering and the Ph.D. degree in electrical engineering from the Polytechnic of Bari, Bari, Italy, in 1994 and 1998, respectively. He was a guest researcher with the Operation Research Institute, Technical University of Aachen, Aachen, Germany, in 1997. Since 1999, he has been an assistant professor of automatic control and technical head of the robotics lab at the Department of Electric and Electronic Engineering, Polytechnic of Bari. He is the author of 75 journal and conference papers. His research interests include computational intelligence and its application to industrial automation and robotics, numerical and combinatorial optimization, discrete-event systems modeling and control, and distributed control of manufacturing systems.

Luigi Delfine received the Laurea degree with honors in electrical engineering from the Polytechnic of Bari, Italy, in April 2002. He has been working with the research group of converters, electrical machines, and drives since November 2002 as a Ph.D. student. His main research interests are focused on intelligent motion control of electrical machines, parameter identification of electrical drives, and autonomous navigation in mobile robotics.

Address for Correspondence: David Naso, Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Re David, 200 – 70125 Bari, Italy. Phone: +39 080 5963 410. E-mail: nasa@poliba.it.