ELSEVIER

Robotics and Autonomous Systems 38 (2002) 1–18

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Sensor-based navigation of a mobile robot in an indoor environment

H. Maaref *, C. Barret

CEMIF—Complex Systems Group, University of Evry, CE 1455 Courcouronnes, 40 rue du Pelvoux, 91020 Evry Cedex, France

Received 14 December 1998; received in revised form 23 May 2001
Communicated by T.C. Henderson

Abstract

The work presented in this paper deals with the problem of the navigation of a mobile robot either in unknown indoor environment or in a partially known one.

A navigation method in an unknown environment based on the combination of elementary behaviors has been developed. Most of these behaviors are achieved by means of fuzzy inference systems. The proposed navigator combines two types of obstacle avoidance behaviors, one for the convex obstacles and one for the concave ones. The use of zero-order Takagi–Sugeno fuzzy inference systems to generate the elementary behaviors such as "reaching the middle of the collision-free space" and "wall-following" is quite simple and natural. However, one can always fear that the rules deduced from a simple human expertise are more or less sub-optimal. This is why we have tried to obtain these rules automatically. A technique based on a back-propagation-like algorithm is used which permits the on-line optimization of the parameters of a fuzzy inference system, through the minimization of a cost function. This last point is particularly important in order to extract a set of rules from the experimental data without having recourse to any empirical approach.

In the case of a partially known environment, a hybrid method is used in order to exploit the advantages of global and local navigation strategies. The coordination of these strategies is based on a fuzzy inference system by an on-line comparison between the real scene and a memorized one. The planning of the itinerary is done by visibility graph and A* algorithm. Fuzzy controllers are achieved, on the one hand, for the following of the planned path by the virtual robot in the theoretical environment and, on the other hand, for the navigation of the real robot when the real environment is locally identical to the memorized one.

Both the methods have been implemented on the miniature mobile robot Khepera® that is equipped with rough sensors. The good results obtained illustrate the robustness of a fuzzy logic approach with regard to sensor imperfections.

Keywords: Mobile robot; Reactive navigation; Fuzzy inference systems; On-line optimization

1. Introduction

Various methods for controlling mobile robot systems have been developed which are generally classified into two categories: global planning and local control. Many works, based on the complete knowledge of the robot and the environment, use a global planning method such as artificial potential fields [11], connectivity graph, cell decomposition [12], etc. These methods build some paths (set of sub-goals) which are free of obstacles. Their main advantages are to prove the existence of a solution

* Corresponding author. Tel.: +33-01-6947-7554; fax: +33-01-6947-7599.
E-mail address: maaref@cemif.univ-evry.fr (H. Maaref).

0921-8890/02/$ – see front matter
PII: S0921-8890(01)00165-8

which permits the robot to reach its destination and to generate collision-free map-making. Thus, in this map, a global optimal solution can be achieved with the assistance of a cost function. The latter is related to either the global route between a start position to a goal position due to the A* algorithm, e.g., the time path, or the security of the mission [18]. However, they have some well-known drawbacks. For example, an exact model of the environment is needed which unfortunately cannot be defined in most applications. Then, it is difficult to handle correctly a modification of the environment due to some new or dynamic objects.

The local methods are mainly used in an unknown environment. They could be called reactive strategies and are completely based on sensory information. Therefore, an absolute localization is not requisite and only the relative interactions between the robot and the environment have to be assessed. In these circumstances, a structural modeling of the environment is unnecessary, but the robot has to acquire through its sensory inputs a set of stimulus–response mechanisms. In this scheme, the robot is generally expected to carry out only simple tasks. Numerous methods have been proposed [4]. They do not guarantee a solution for the mission because of the occurrence of deadlock problems. The reason is that the robot does not have a high-level map-reading ability. For more efficiency and safety, perception tools have to be increased (several types of sensors including, e.g., cameras) to get more pertinent information about the environment. But then it is not easy to process the data under real time constraints. These constraints often lead to a degradation of the accuracy and the richness of the information.

Some constraints are added to the intrinsic drawbacks of these methods caused by:

- the imprecision or lack of knowledge in understanding all the phenomena contributing to the behavior of the system and its environment;
- the difficulties to represent correctly the environment and to locate the robot, due to errors in the sensors data which are still far from perfect, taking into account the present day technologies.

In other respects, a set of methodologies, called qualitative or approximate reasoning, have been developed to build a decision making approach in systems where imperfection cannot be completely avoided or corrected. These methodologies attempt to capture some aspects of the human behavior in system control. Their aim is to incorporate implicitly the imperfection in the information gathering and reasoning process, rather than to determine them explicitly through numerical calculations or mathematical representations.

Some qualitative reasoning theories have been developed over the past few years [10] and currently the most used for application in control systems is the theory of fuzzy sets [30]. The control based on this theory [13] provides satisfying results even in cases where classical control failed. As a fuzzy controller is built following the knowledge of experts, a complex or ill-defined system can be described without using an exact mathematical model. Therefore, the fuzzy sets theory is a good candidate both to handle imprecision and to assign built-in guidance control enabling the robot to navigate throughout complex environments. In fact, we know from our own experience of human motion that it is unnecessary either to know our own exact location or to have a comprehensive knowledge of the whole scene. It can be sufficient, e.g., to know whether there is enough free space to get around an obstacle and to recognize marks indicating whether the passageway leads to the goal or not. Many application works of fuzzy logic in the mobile robot field have given promising results [23,27,28], etc.

The finality of our work consists of developing low cost navigation strategies in indoor environment, e.g., the aim is to help disabled people [8]. In this context, the main concern is to build efficient navigation techniques giving more priority to safety than to optimality. Fig. 1 gives a global scheme of the adopted strategy. It is based on the fact that generally one can dispose of a building's map in which some main fixed elements of the environment are located: walls, doors, heavy and fixed furniture, etc. But, many unfixed elements, whose positions is a priori unknown, can be added to the initial map. In this situation, two extreme cases can happen. If the environment detected by the robot corresponds to the memorized map, then the robot should follow with high speed a planed trajectory using a global method. On the contrary, if the environment is not recognized, a displacement at a reduced speed has to be generated by a local method of reactive navigation. Between these two extreme

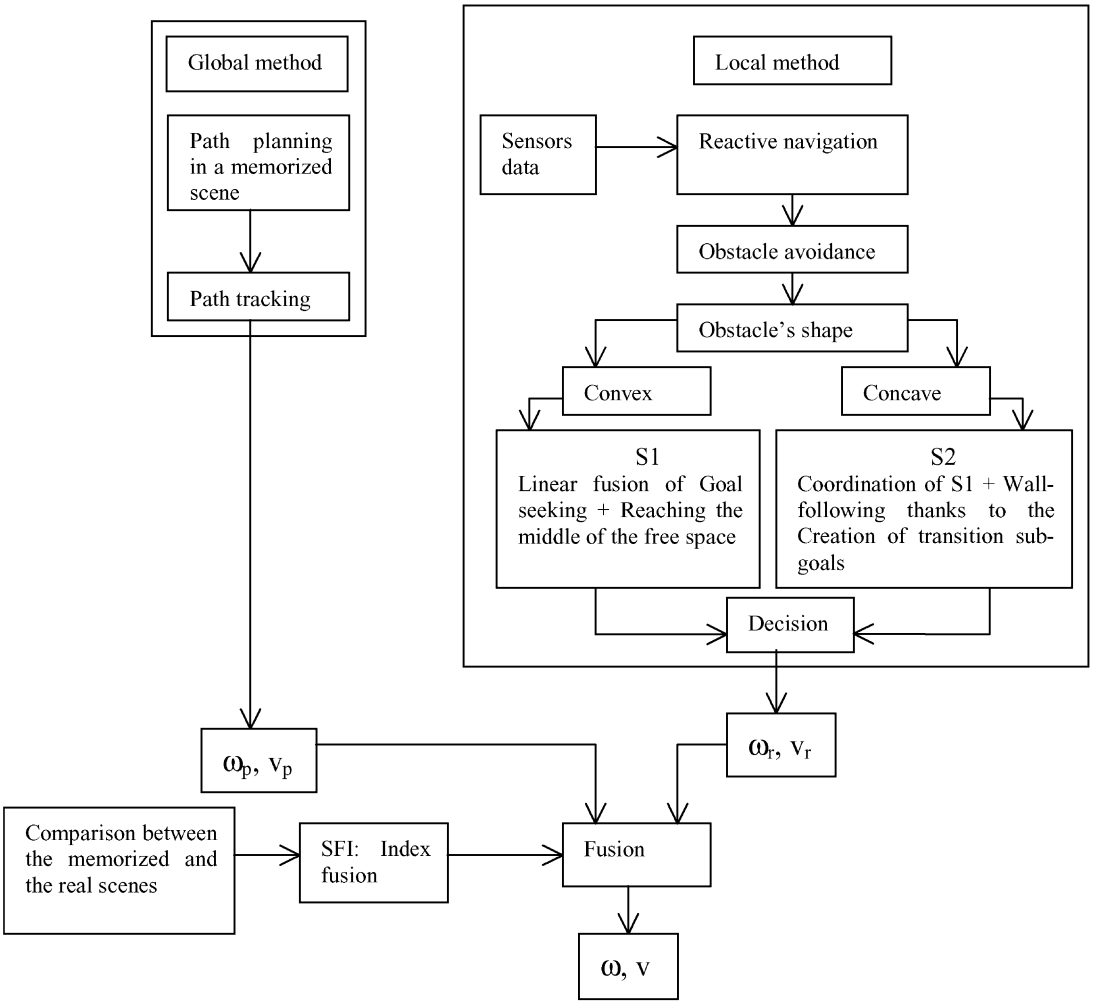


Fig. 1. Global scheme of the adopted strategy.

situations, a progressive evolution must be done by fusing outputs coming from both modules as a function of a degree of recognition of the memorized scene.

This paper is organized as follows: first the used mobile robot is described and some working assumptions are given in Section 2. Section 3 presents the local method for navigation in an unknown environment. In Section 4 the global method used in known environment is given and the fusion of both the methods is developed. Finally, a conclusion is given in Section 5.

2. Physical implementation and working assumptions

The experimentation is mainly done on Khepera® which is a small mobile robot developed at the Ecole Polytechnic Fédérale de Lausanne (EPFL). Our motivations to work with such a miniature robot are the following:

1. Our methodology is based on developing strategies using logical rules independently of a precise model of the robot. So the transfer of control

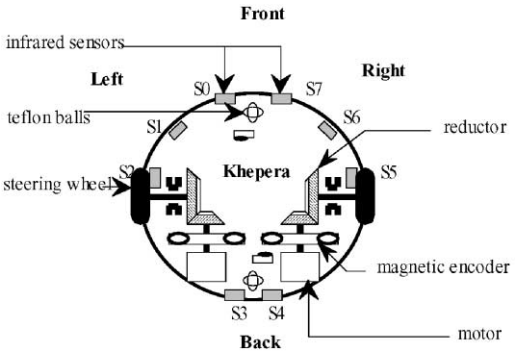


Fig. 2. The miniature mobile robot Khepera®.

algorithms from one robot to another is not a difficult problem.

2. Nevertheless, to work with a real robot is largely preferable to use simulations as far as, e.g., dealing with sensor imperfections or real time constraints is concerned.
3. Finally it is clear that the easiness to build and modify the environment of a mini robot is greatly appreciable.

Khepera® has a circular shape featuring 55 mm in diameter ($2r$), 30 mm in height and 70 g in weight [20]. Two wheels and two small Teflon balls support it. The robot possesses eight infrared sensors, which are composed of an emitter and an independent receiver. These sensors $(S0, S1, \ldots, S7)$ are disposed in a somewhat circular fashion around its body (Fig. 2) and allow the measurement of distances in a short range from about 1 to 5 cm. Its maximum linear speed is about 40 mm/s.

The robot's linear and angular speeds are sent from a host computer via a serial link to an on-board chip, which is based on a Motorola 68331 micro-controller. The linear speeds of the right and left wheels are then calculated.

In this study, we assume the following conditions:

- The robot moves on a flat ground.
- Inertial effects are neglected.
- The used mobile robot has the non-holonomic characteristic but this later is not constraining.
- The robot moves without sliding and can be localized when it finds itself in a locally known scene [22].

Most of the experiments are done on both the real and a simulated mobile robot. The simulator dedicated to Khepera® has been written in C++ by Michel [19] and runs on SUN Sparc station. The experimental results deduced from the real and simulated mobile robot are very near.

3. Navigation strategies in unknown environment

3.1. Principle

In a totally unknown environment, the navigation is done completely in a reactive manner. So a classical method such as the artificial potential fields [11] could be used. But it is well known that this method suffers from local minima problems leading to blocking situations. A solution has been proposed in a previous work [14] based on an automatic tuning of attractive and repulsive force coefficients due to fuzzy rules. Nevertheless some oscillation problems remain in narrow environments and passageways, which are very constraining for dedicated utilities indoor robotics.

The described approach (Fig. 1) here is largely based on fuzzy inference systems (FISs) and inspired from human behavior, which consists to reach the free space while seeking the goal (strategy S1). This allows avoiding local minima by reaching the middle of the available free space when the robot passes through a cluttered environment [2]. But some failing situations are yet encountered in the case on concave obstacles. That is why coordination of S1 and another elementary behavior of wall-following type including the creation of transition sub-goals develop a second strategy S2. As a matter of fact, the idea is to anticipate in order to avoid a potential blocking situation rather than to discover it and subsequently react. So, an obstacle will be in fact qualified as *concave* if all the used exteroceptive sensors give simultaneously small measurements of distances, since, even if the obstacle has not really a concave geometric shape, it is preferable to trigger the S2 strategy instead of taking the risk to fall in a blocking situation with S1 strategy.

To skirt the two sides of the wall, the detection of a concave obstacle (Fig. 3) provokes the creation of an intermediate sub-goal of transition "SG[i]" at the point of detection and triggers the wall-following behavior to act, e.g., on the left side. If the robot goes

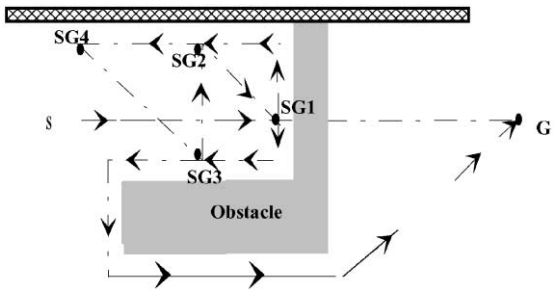


Fig. 3. Concave obstacle skirting.

away from the target and the distance of displacement is greater than a threshold distance T; it turns back to the intermediate sub-goal SG[i] previously memorized, due to the strategy S1. Then, it skirts the obstacle on the other side, with the same threshold distance T. The wall-following ceases if the two following conditions are filled:

- The three sensors measure big distances.
- The goal is in the right or in the left (depending on the side of the obstacle followed by the robot) quadrant with respect to the actual direction of the robot.

The developed algorithm allows a robot with exteroceptive sensors to travel from any start point S to any target point G in a cluttered environment without any prior knowledge on the location of the obstacles.

3.2. On-line optimization of FISs for reactive strategies

The reactive strategies of navigation (reaching a collision-free space, goal-seeking and wall-following) are completely based on sensory information. Two of them (reaching a collision-free space and wall-following) are built due to self-tunable fuzzy inference systems (STFISs) controlling the angular ω and linear v speeds of the mobile robot. The angular speed is generated first at a given linear speed and, then after convergence of this later structure, the control rules of the linear velocity are deduced.

With respect to the use of a classical, manually tuned FIS to build the reactive behaviors of the robot, the STFIS has the following two main advantages:

- It avoids the manual tuning of the parameters of the FIS that can be in some cases quite long and cumbersome. Moreover, this manual tuning leads inevitably to a sub-optimal behavior.
- It allows to cope exactly with the physical characteristics of the robot. If either these characteristics evolve with time or the robot is changed (or a change from a simulator robot to a real one is carried out), the controller will adapt automatically to the new situation.

The structure of the FIS is as follows. The membership function for the input values are triangular and fixed. A min operator performs the conjunction of the inputs and the conclusions of the rules are numerical values W_i (so-called weights). They are optimized through a learning process [1].

The shape of the used membership functions is triangular and fixed in order to extract and represent easily the knowledge from the final results. So the output value y (v or ω) is given by

$$y = \frac{\sum_{i=1}^{n} W_i \times \alpha_i}{\sum_{i=1}^{n} \alpha_i},$$

where α_i are the truth values of each fired rule.

The learning architecture is presented in Fig. 4. This architecture is a simplified version of the "distal

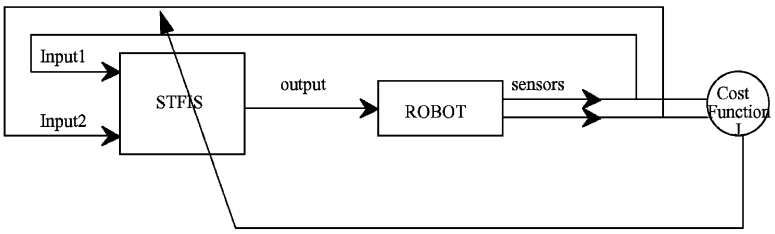


Fig. 4. Learning architecture.

control" method proposed by Jordan and Rumelhart [9] for neuro-control. In the original method, two neural networks are used: one for modeling the plant and another for the controller. In fact, as pointed by Jordan and Rumelhart it is not necessary to work with an accurate model of the plant to obtain an efficient control. Saerens [26] and Renders [24] have shown that the model network can be successfully approximated by the sign of the terms of the Jacobian matrix of the plant (in the assumption that these signs are fixed on the working space, which is valid for a lot of real systems). These results have been extended by substituting to the neural controller a fuzzy controller with adaptive parameters [5], leading to the very simple architecture as in Fig. 4 for single input single output (SISO) systems.

The learning is entirely done on-line on the actual robot. The table of rules (weights W_i) is initially empty. The robot acquires by its sensors the distances to the environment, calculates the error to be back propagated, updates the triggered rules in real time, begins to move and so on, etc. The weights of the table of decision are then adjusted locally and progressively. As the learning progresses, the mobile is more and more able to cope with new situations.

The back-propagation training technique [25] updates weights according to:

$$W(k+1) = W(k) + \eta\left(\frac{-\partial J}{\partial W}\right),$$

where k is the training iteration, J is the cost function used in the learning algorithm, η is the learning rate and $\Delta W(k) = W(k) - W(k-1)$.

If the classical quadratic error is used as a cost function, $J = \frac{1}{2}\varepsilon^2$ where ε depends on the task; the back-propagation minimizes effectively the value of J, leaning rapidly to a good reactive navigation. But, if the learning is prolonged, the weights increase continuously with time and, progressively, the quality of the control decreases. To overcome this difficulty, a technique known as "weight decay" in classification methods [6] and having a strong relation with ridge regression and regularization theory [3] is used. So a second term is included in the cost function that becomes

$$J = \frac{1}{2}\varepsilon^2 + \lambda \sum W_i^2,$$

where λ is a coefficient proportional to $\alpha_i / \sum \alpha_i$. It is chosen so that the output value does not exceed the maximum angular speed of each wheel of the robot (1.58 rad/s). By applying this method, a saturation of the growth of the weights is obtained without any degradation of the residual quadratic error and the quality of the control is maintained even under prolonged learning.

3.3. Avoidance of convex obstacles

This navigator is built by fusing two elementary behaviors: a self-tunable fuzzy controller to reach the middle of the free space and a crisp one to track the current sub-goal.

3.3.1. Reaching the middle of the collision-free space behavior

When the vehicle is moving towards the target and the sensors detect an obstacle, an avoiding strategy is necessary. The method consists of reaching the middle of a collision-free space. This behavior is obtained by means of an STFIS.

The input variables are respectively the normalized measured distance on the right (R), on the left (L) and in front (F) such as

$$R_n = \frac{R}{R+L}, \quad L_n = \frac{L}{R+L}, \qquad F_n = \frac{F}{\sigma},$$

where front data $F = \min(\mathrm{S0}, \mathrm{S7})$; right data $R = \min(\mathrm{S6}, \mathrm{S7})$; left data $L = \min(\mathrm{S1}, \mathrm{S2})$ and σ is a distance beyond which the obstacles are not taken into account. Due to this normalization, the universes of discourse evolved automatically with the sensor data (Fig. 5).

The shape of the membership function is triangular and the sum of the membership degrees for each variable is always equal to 1. The universes of discourse are normalized between 0 and 1.

For this behavior and to generate first the control rules for the angular speed ω_a, the error used in the cost function is given by $\varepsilon_\omega = Y - \frac{1}{2}(Y + F_n)$ where Y is either R_n or L_n. After a few rounds at a constant linear speed on a learning track, the navigation of the robot is satisfying.

The weights of the controller converge to the values given in Table 1, where the linguistic labels for the inputs are defined as: Z (zero), S (small), M (medium),

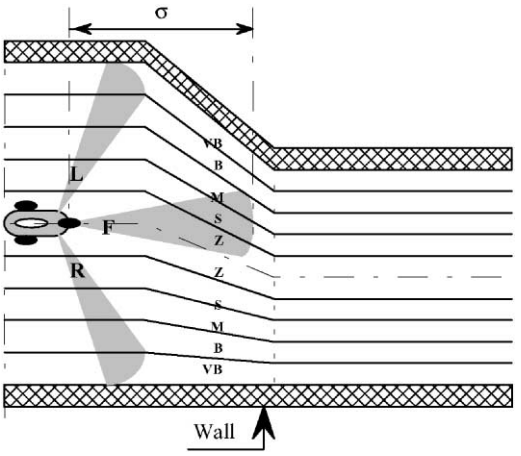
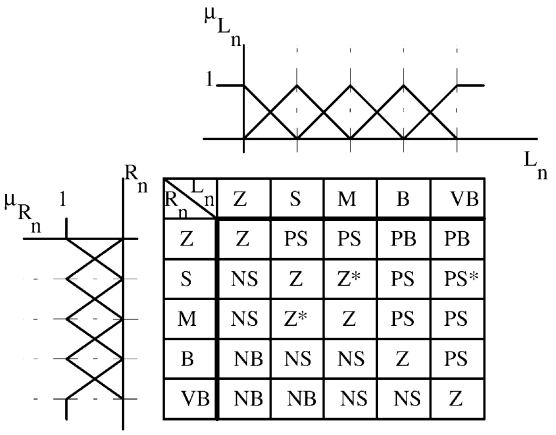


Fig. 5. Evolution of the universe of discourse with the width of the environment.

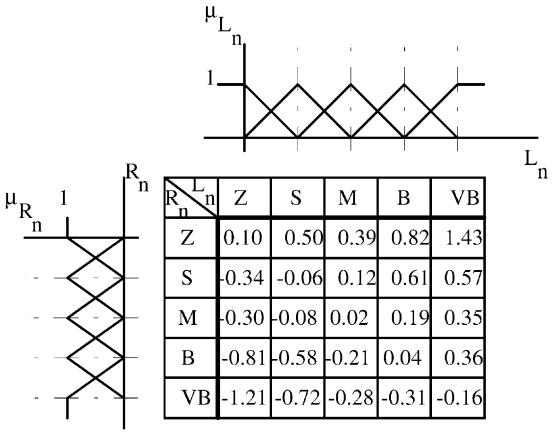
B (big) and VB (very big). These numerical values could be eventually translated in symbolic values to verify the logical meaning of the rules. We can assign to them a linguistic interpretation by substituting the symbolic concept PB (positive big) for the values greater than 0.7, PS (positive small) for the values between 0.2 and 0.7, Z (approximately zero) for the values between −0.2 and 0.2, NS (negative small) for the values between −0.2 and −0.7, and NB (negative big) for the values lesser than −0.7. We obtain the linguistic table for the angular speed from Table 2. It is interesting to compare this later with a table written empirically from experience of a human driver, and following the very usual diagonal structure known as McVicar–Whelan's [17] controller (Table 3). We can observe that the two linguistic sets of rules are very near. Only three cases (noted with ∗) are different and they differ from only one linguistic concept (PS instead of PB and Z instead of PS and NS). So, we can claim that the extracted rules are quite logical and coherent. Moreover, the use of STFISs allows the optimization of the controller with respect to the actual characteristics of the robot. This means that the rough and manual tuning of the parameters of the fuzzy controller is replaced by a fine local automatic tuning and

Table 2
Linguistic table for the angular speed



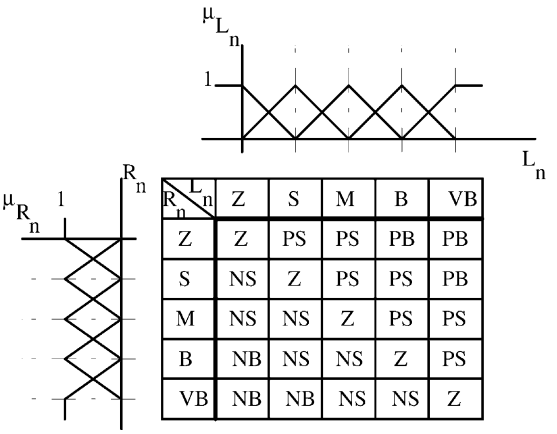
R_n \ L_n	Z	S	M	B	VB
Z	Z	PS	PS	PB	PB
S	NS	Z	Z*	PS	PS*
M	NS	Z*	Z	PS	PS
B	NB	NS	NS	Z	PS
VB	NB	NB	NS	NS	Z

Table 1
Angular speed coefficient rules



R_n \ L_n	Z	S	M	B	VB
Z	0.10	0.50	0.39	0.82	1.43
S	-0.34	-0.06	0.12	0.61	0.57
M	-0.30	-0.08	0.02	0.19	0.35
B	-0.81	-0.58	-0.21	0.04	0.36
VB	-1.21	-0.72	-0.28	-0.31	-0.16

Table 3
Linguistic table deduced by human expertise



R_n \ L_n	Z	S	M	B	VB
Z	Z	PS	PS	PB	PB
S	NS	Z	PS	PS	PB
M	NS	NS	Z	PS	PS
B	NB	NS	NS	Z	PS
VB	NB	NB	NS	NS	Z

this can improve very significantly the performances, e.g., a given way is traveled more quickly with the STFIS controller than with the classical controller by taking into account the actual maximum speed of the robot's wheels.

A structure of the same type is used to generate the control rules for the linear speed v_a as a function of the angular speed ω_α and the front distance F. The cost function is realized with

$$\varepsilon_v = 40 - \max(|v_a + r\omega_\alpha|, |v_a - r\omega_\alpha|) - (1 - \tfrac{1}{5}F) \cdot 40.$$

This allows to attain the maximum speed (40 mm/s) and to decrease the speed as a function of F.

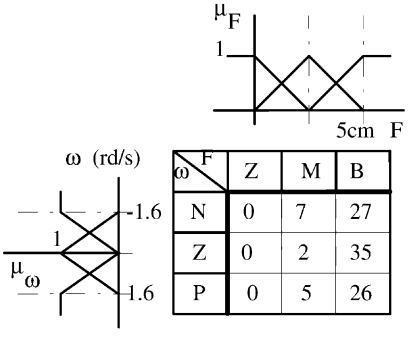
The linguistic labels for ω are defined as N (negative), Z (approximately zero) and P (positive) and for F they are Z (approximately zero), S (medium) and B (big). The output weights of the controller after learning are given in Table 4.

It is easy to verify that these weights correspond rules expressing that the more the robot has to turn and the closer a frontal obstacle is, the greater is the reduction of the linear speed. Fig. 6 presents an example of navigation in a real cluttered environment. The self-tunable fuzzy controller shows its efficiency to realize the task. But in order to reach its goal the robot has to be provided with a goal-seeking behavior.

3.3.2. Goal-seeking behavior

The basic scheme is given in Fig. 7. The goal G produces an attractive force F_a that guides the robot to its destination. The actions ($C_{\omega g}$ and C_{vg}) generated by this force are modulated by the inverse of the distance

Table 4
Linear speed coefficient rules



ω \ F	Z	M	B
N	0	7	27
Z	0	2	35
P	0	5	26

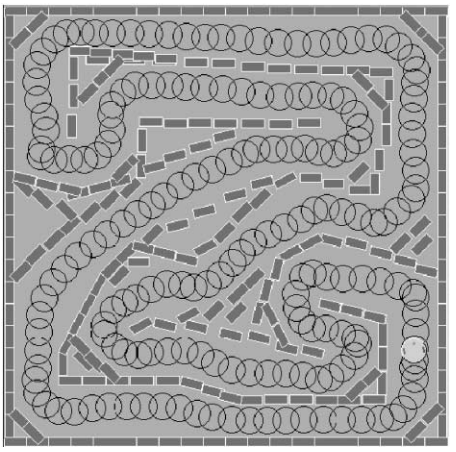


Fig. 6. "Reaching the middle of the collision-free space" behavior: experimentation with the simulator.

PG between the center of the robot and the goal. θ_g is the angular deviation needed to reach the goal. D is the distance of influence of the goal. It is supposed that no obstacle exists in the circle of diameter D.

When the robot is far enough from the sub-goal ($PG > D$) the angular speed coefficient is given by

$$C_{\omega g} = \frac{C_g}{PG}\frac{D}{\pi}\theta_g.$$

The coefficient C_g is chosen in such a way that the robot reaches a maximum angular speed for $\theta_g < \pi$. So it does not deviate too much from the PG direction. As soon as the robot reaches the influence zone of the goal ($PG < D$) the angular speed coefficient

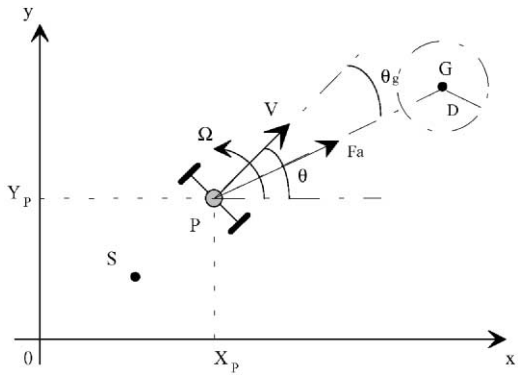


Fig. 7. Goal-seeking scheme.

becomes

$$C_{\omega g} = \frac{C_g}{\pi}\theta_g.$$

In both the cases $C_{\omega g}$ is normalized so that $|C_{\omega g}|$ cannot exceed 1. Moreover, the goal-seeking linear speed coefficient is determined in relation to $C_{\omega g}$ by the equation

$$C_{vg} = 1 - |C_{\omega g}|.$$

This expresses the following rule: the more the robot is pointed towards the goal direction or the further the robot is from the goal, the faster it can move (knowing that the speed is bounded by a maximal value either by the user or by the hardware).

3.3.3. Fusion of "reaching of the middle" and "goal-seeking" behaviors

In reactive navigation, the safety of the robot is essential. For this reason, we distinguish two cases:

- If an obstacle is detected very close to the robot, on only one side or in the front, then the obstacle avoidance has priority and the attraction is cancelled ($C_{\omega g} = 0$).
- Else, the angular speed set-point ω_r applied to the robot results from a linear combination between the obstacle avoidance and the sub-goal attraction:

  $$\omega_r = \alpha\omega_a + \beta C_{\omega g}\omega_{max},$$

  where α and β are coefficients adjusted by experimentation to get the best trajectory generation and ω_{max} is the maximum chosen angular speed. The linear speed V_r set-point is given by

  $$V_r = \min(V_a, C_{vg}V_{max}),$$

  if the robot is outside the zone of D radius. Else, it is reduced so that

  $$V_r = \min(V_a, C_{vg}V_{min}),$$

  where V_{max} and V_{min} are the maximum and minimum chosen linear speed, respectively.

An example of implementation of this fusion rule on the robot Khepera® is shown in Fig. 8. The task consists in getting through a doorway in an environment like a flat. For more visual clarity, the obstacle is drawn on the screen in accordance with the sensor impacts. The robot avoids the obstacle while seeking the goals (G1, then G2).

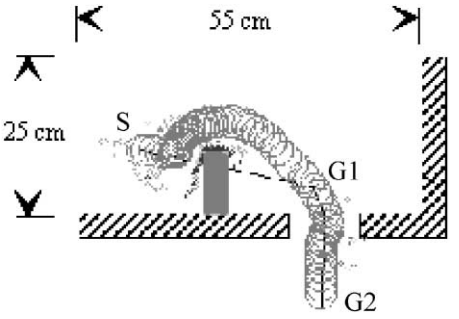


Fig. 8. Avoidance of convex obstacles: experimentation with Khepera®.

3.4. Avoidance of concave obstacles

In an environment composed with concave obstacles and in order to avoid blocking situations, we use an additional behavior, inspired of the myopic method, which consists of following the contour of the obstacle in order to skirt round it. This behavior is built by means of an STFIS. The goal is to follow the walls surrounding the robot at a "d_setpoint" distance, with regard to the sensor measurements: F (front) and L (left) or F and R (right) (Fig. 9).

The shape of the membership functions is triangular and the universes of discourse are defined between 0 and σ (5 cm for Khepera®) for the inputs. For this behavior, the error used in the cost function for the angular speed is given by $\varepsilon_\omega = \min(Y, F) - \text{d\_setpoint}$, where Y is either R (wall-following on the right side) or L (wall-following on the left side) and d_setpoint is a given set-point distance. On the beginning of the learning the robot is near a wall in an unknown

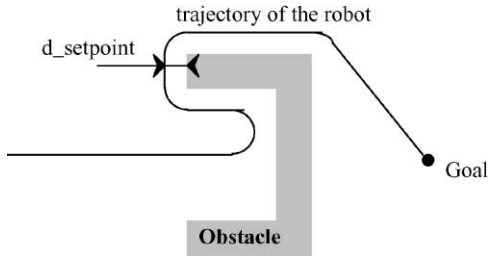


Fig. 9. Wall-following strategy.

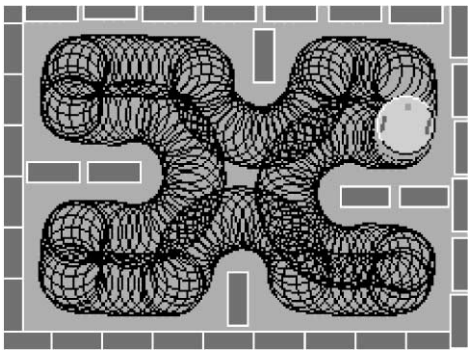
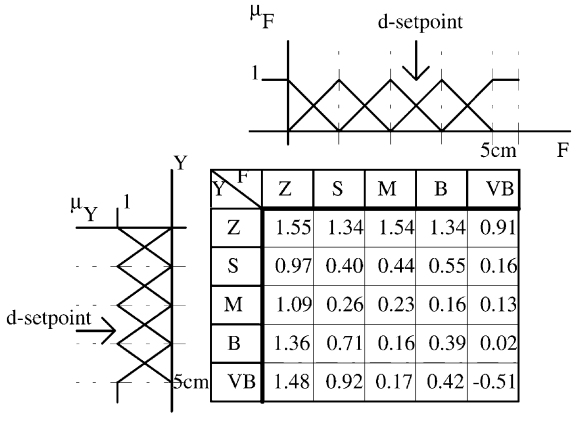


Fig. 10. Wall-following learning track: experimentation with the simulator.

environment. After a few rounds at a constant linear speed on the learning track (Fig. 10), the robot is able to follow all the walls of the track at the given distance.

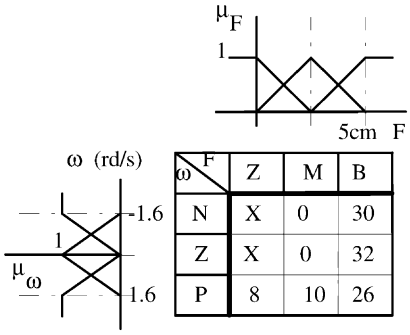
At this time, the output weights of the controller have converged to the values given in Table 5 where the linguistic labels for the inputs are defined as: Z (zero), S (small), M (medium), B (big) and VB (very big). For the linear speed, the structure is the same one as for the "reaching the middle of the collision-free space" behavior. After convergence, the obtained numerical values are given in decision Table 6. The logical meaning of the rules is obvious since they verify that the more the angular speed increases and the closer a frontal obstacle is, the greater the reduction of the linear speed is. The blocks marked with the symbol X are never triggered because, if the robot turns on the right, that's means there is no wall in front.

Table 5
Decision table for angular speed (rad/s)



Y \ F	Z	S	M	B	VB
Z	1.55	1.34	1.54	1.34	0.91
S	0.97	0.40	0.44	0.55	0.16
M	1.09	0.26	0.23	0.16	0.13
B	1.36	0.71	0.16	0.39	0.02
VB	1.48	0.92	0.17	0.42	-0.51

Table 6
Decision table for linear speed



ω \ F	Z	M	B
N	X	0	30
Z	X	0	32
P	8	10	26

The robot is now able to follow correctly over the walls of the any shape at the given set-point distance with a smooth and continuous trajectory (Fig. 11). The whole algorithm for concave obstacle avoidance has been tested on the robot Khepera®. In Fig. 12(a), only one sub-goal is created, because the value of the threshold of displacement T is quite big ($T = 1\,\text{m}$). In Fig. 12(b), the threshold T is smaller ($T = 0.5\,\text{m}$): three intermediate sub-goals are created now before the robot converges towards the final goal. Besides, T is chosen depending on the environment size and constraints of the mission. As a general rule, too low a

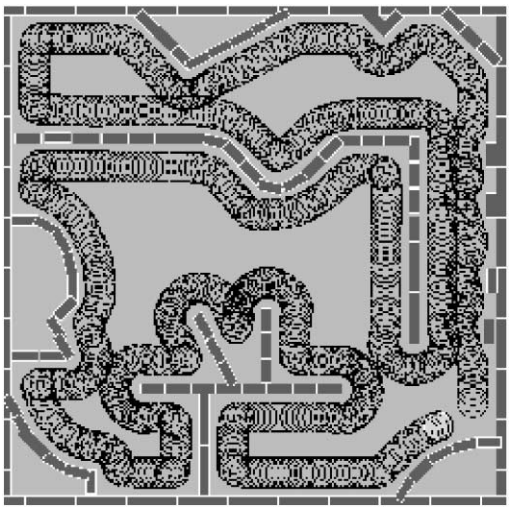


Fig. 11. Wall-following generalization track.

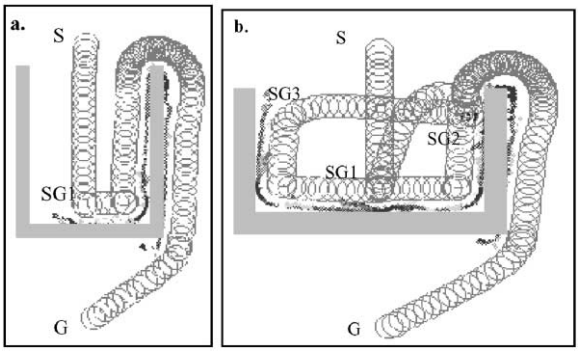


Fig. 12. Experiments of concave obstacles skirting with Khepera®.

value of T provokes many direction changes, increasing the imprecision of the localization. In the opposite case, too high a value mainly leads to sub-optimal trajectories.

3.5. Coordination of behaviors

Now the whole strategy of reactive navigation (as described in Fig. 1 and Section 3.1) in a complex environment, using all the developed reactive agents, can be applied. An example of result is shown in Fig. 13 where the robot avoids and skirts successfully obstacles of various shapes before to attain its goal. In fact, the S2 strategy is activated in "Z1" and "Z3" zones by coordination of the S1 strategy and the wall-following behavior due to the creation of an intermediate sub-goal. In "Z2" and "Z4" zones only the S1 strategy is triggered.

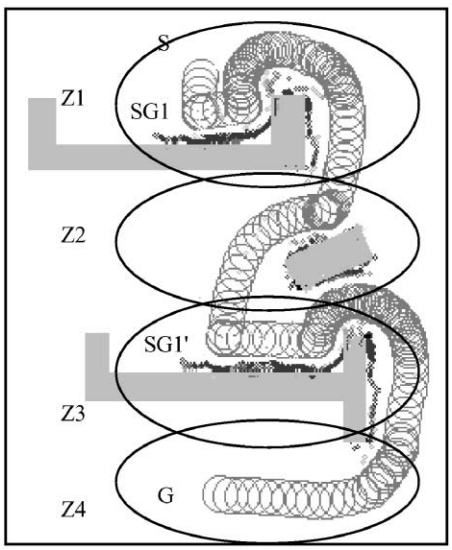


Fig. 13. Coordination of behaviors.

4. Navigation in a partially known environment

In the case of indoor robotics field, one has to exploit the a priori knowledge of the environment that takes the form of the map containing the main characteristic features (walls, doors, fixed furniture, etc). So it obvious that an efficient control of the mobile robot needs:

- a local level based completely on the information of different sensors covering the close circle of the vehicle;
- a high-level for path planning using a global description of the world with possibly incomplete and/or imperfect knowledge.

The original idea is to keep in memory this pre-acquired knowledge contrary of most works done in this field [12], where the a priori knowledge is used only to generate sub-goals. This allows having a safe navigation, to modulate continuously the speed and eventually to update the map.

The approach exploits that the a priori knowledge on the environment (scene called "memorized" in which a virtual robot moves) which is susceptible to local variations by modification and/or by addition of obstacles (scene called "real") (Fig. 14).

4.1. Planned path following in a real known environment

For the planning of a path, the visibility graph and the A* algorithm are used. The visibility graph [12] is a set of straight lines connecting the source, the goal and obstacle vertices. Each point is connected to all viewed points without intersecting obstacles (Fig. 15). Then, an optimal path is searched with an A* algorithm in the generated graph, using the Euclidean distance as a cost function. This path is a polygonal line connecting the source to the goal; it is the shortest collision-free path from source to goal.

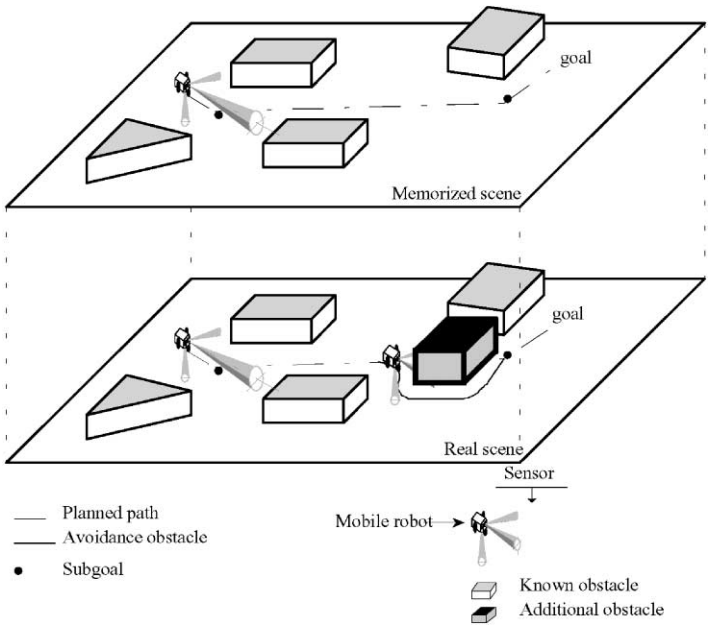


Fig. 14. Comparison between real and memorized scenes.

This method is well adapted to generate a path (set of sub-goals) for a robot represented by a point. In order to consider the whole ground space occupied by the robot, we need to extend the area of the obstacles. In our case the used robots have circular shape. Then, the obstacles are dilated by a distance equal to the diameter of the robot with a revolution symmetry such as the arcs of the circle are approximated by some segments.

In Fig. 16 we show an example of environment for the mini robot Khepera®. The obstacles are the shaded polygons. They are surrounded by a dotted line representing the dilatation. The optimal path obtained by the A* algorithm is the dashed line joining the source point to the goal point through some sub-goals indicated by the black points.

The path to follow is the segments joining the successive sub-goals. In order to assure the control

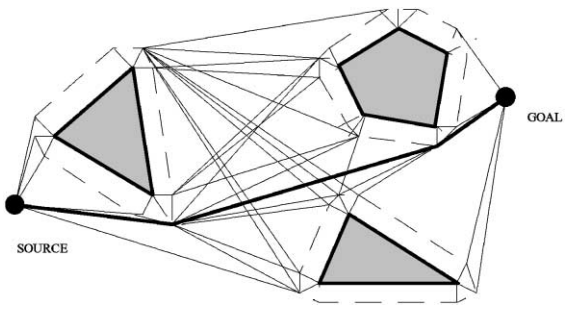


Fig. 15. Optimal path.

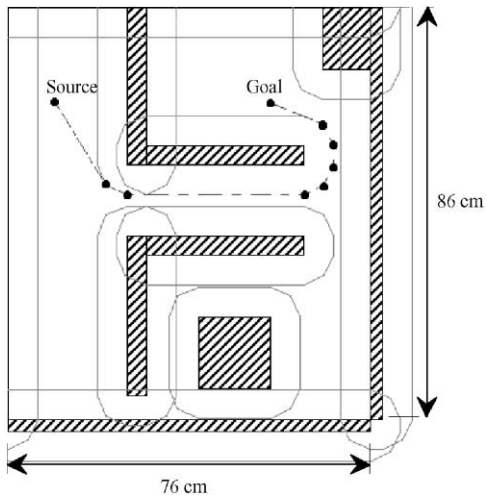


Fig. 16. Path planning in a real environment.

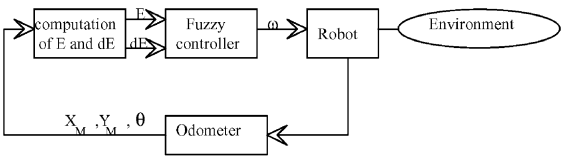


Fig. 17. Control architecture for the path tracking.

of the robot between these sub-goals, various methods can be used. These methods can use classical [7,29], etc. or fuzzy control [21,31], etc. The module of control developed here to generate the path between the sub-goals is based on a classical fuzzy control (Fig. 17). It provides the angular speed (ω_p) of the robot which is supposed to evolve at a given linear speed (v_p). The angular speed of the robot is determined from its current position with regard to the path and is achieved by the relative variation of angular speeds of driving wheels. Since the robot is an indeformable solid, the knowledge of the distance E (between the center point M of the robot and the segment joining the sub-goals D and A (Fig. 18)) and of the variation of this distance is sufficient to achieve the task.

The co-ordinates (X_M, Y_M, θ_M) of the robot are given by odometry. The signed distance E is given by

$$MH = E = DM \sin(\widehat{ADM}) = \frac{P}{DA},$$

with

$$P = (X_M - X_D)(Y_A - Y_D) - (X_A - X_D)(Y_M - Y_D).$$

The controller is constituted of a set of fuzzy rules is given in Table 7. The signification of the used linguistics terms is the same as in Section 3.3.

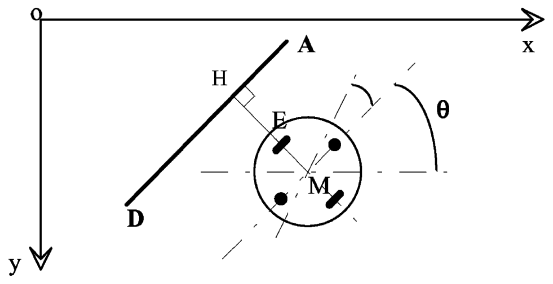
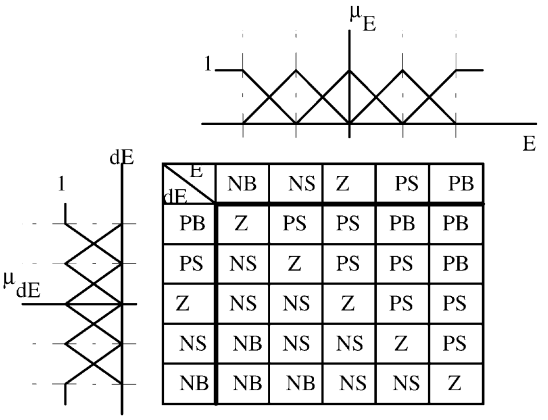


Fig. 18. Position of the robot with regard to the path.

Table 7
Rules of the path tracking fuzzy controller



E / dE	NB	NS	Z	PS	PB
PB	Z	PS	PS	PB	PB
PS	NS	Z	PS	PS	PB
Z	NS	NS	Z	PS	PS
NS	NB	NS	NS	Z	PS
NB	NB	NB	NS	NS	Z

The membership functions are of triangular shape, on a normalized universe of discourse between -1 and 1. The operators used in the FIS are similar to those appearing in a Mamdani controller [16]: min for the composition of the input variables and for the fuzzy implication and max for the aggregation of the rules. The center of gravity method is used for the defuzzification, in order to determine the crisp output actions.

Fig. 19 represents an example of experimental result with the whole previously described method (path planning by visibility graph and A* algorithm and path tracking by the fuzzy controller) when the real scene is identical to the memorized one.

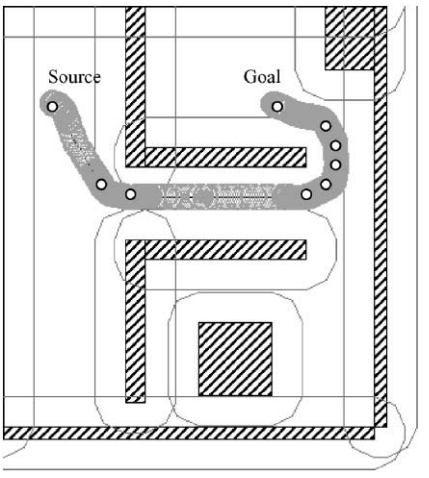


Fig. 19. Displacement of the robot in a known scene.

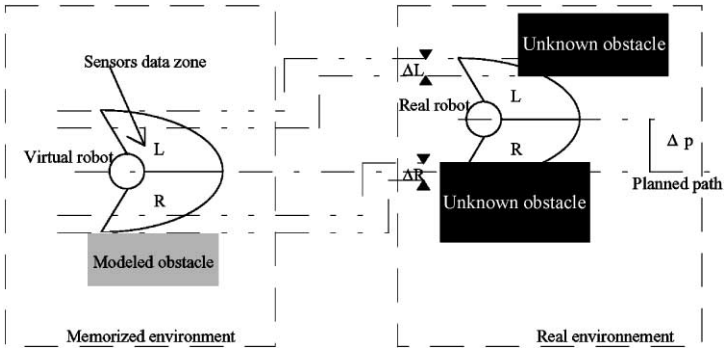


Fig. 20. Comparison of sensors data.

4.2. Fusion of reactive and planed navigation

The aim of this procedure is to navigate the robot from the initial point till the target point by following as nearly as possible the optimal way without taking into account the missing obstacles and avoiding the unexpected obstacles. For this, a virtual robot displacing in the memorized scene and equipped with two lateral virtual sensors is used. An index of preference, indicating which command is the best to apply, strategies fusion index (SFI), is generated by a fuzzy decision making module, the inputs of which are the difference between (Fig. 20):

1. Each sensor data in the memorized scene (modeled as perfect sensor) and the corresponding one in the real scene (data with error) (ΔL, ΔR). One can note that the comparison of the two lateral sensor data is sufficient to accomplish the task.
2. Absolute positions of the mobile robot in the memorized and the real environments (Δp), knowing that the virtual robot moves along with the orthogonal projection on the planned path of the center point M of the real robot.

The SFI value reflects the situation of the robot with respect to the known environment. By exploiting it, the two strategies (global and local) are fused, by weighting of the orders such as

$$\omega = \omega_p \times \mathrm{SFI} + \omega_r \times (1 - \mathrm{SFI}),$$
$$v = v_p \times \mathrm{SFI} + v_r \times (1 - \mathrm{SFI}),$$

where ω and v are the orders to apply to the robot.

Thus, if the sensor data in the two scenes are very close, the navigation in the real scene will be made by tracking the planned path. If they are completely

Table 8
Rules table for SFI

Δ p \ Δ R / Δ L	N/N	N/Z	N/P	Z/N	Z/Z	Z/P	P/N	P/Z	P/P
Z	Sr	Sr	Sr	Sr	**Sp**	**Sp**	Sr	**Sp**	**Sp**
M	Sr	Sr	Sr	Sr	Sr	**Sp**	Sr	**Sp**	**Sp**
B	Sr	Sr	Sr	Sr	Sr	Sr	Sr	Sr	**Sp**

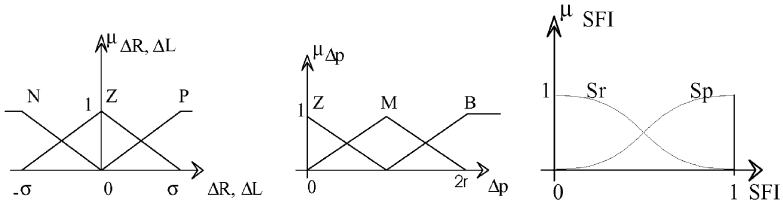


Fig. 21. Fuzzy subsets for the variables ΔR, ΔL, Δp and SFI.

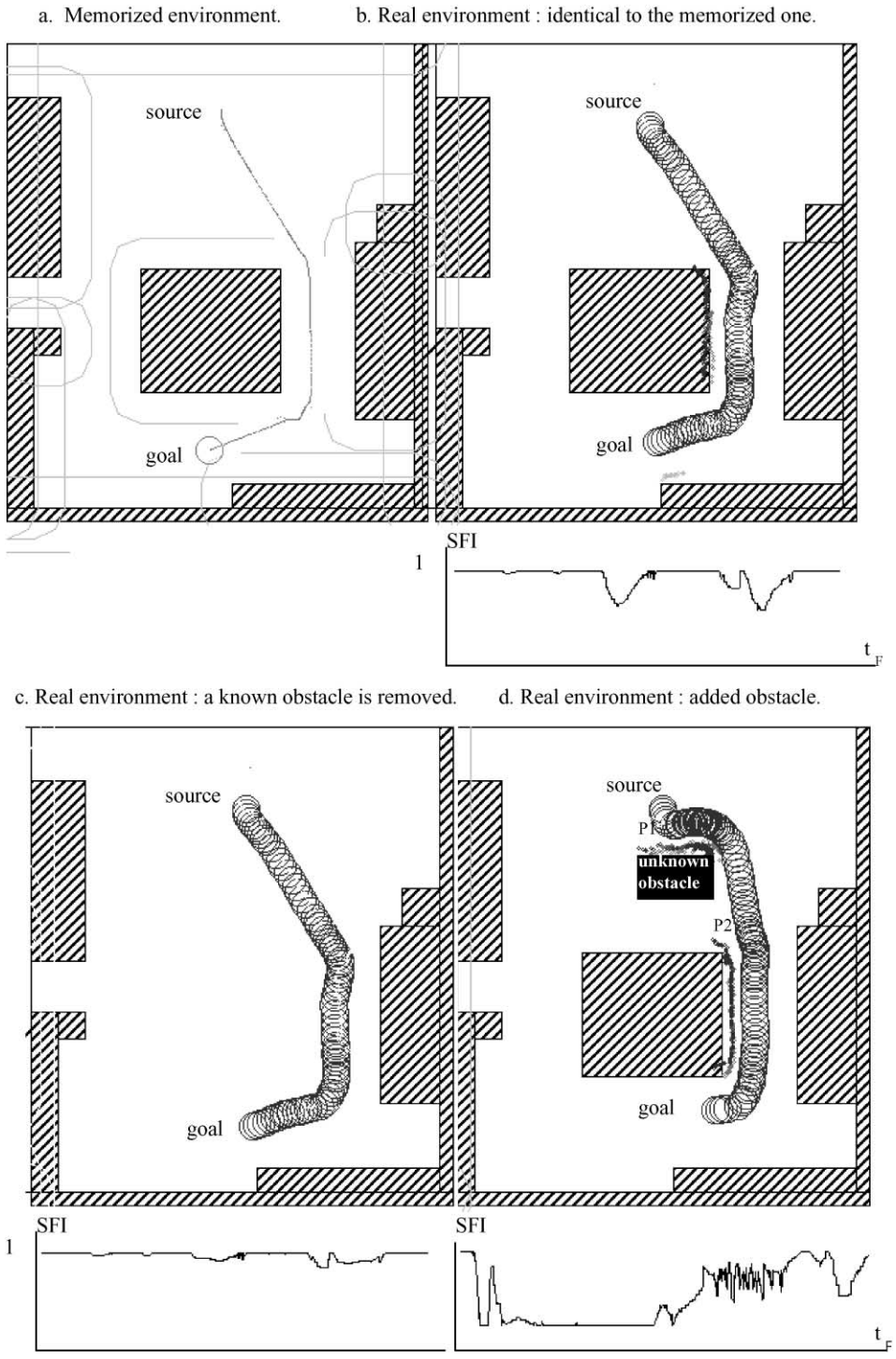


Fig. 22. Examples of experimental results: (a) memorized environment; (b) real environment (identical to the memorized one); (c) real environment (a known obstacle is removed); (d) real environment (added obstacle).

different a strategy of local navigation is triggered. The variable SFI is the output of a fuzzy module and is shared in two fuzzy subsets labeled Sp for releasing the planned path following and Sr for starting the reactive navigation. The universes of discourse of the input variables ΔR, ΔL and Δp are composed of three fuzzy subsets (Fig. 21). The used labels are N (negative), Z (zero), P (positive), M (medium) and B (big).

The fuzzy rules have the following form:

- If ΔR, ΔL and Δp are zero then the planned path following strategy is activated (Sp).
- If ΔR and ΔL are negative and Δp is big then the local navigation is triggered (Sr).

A rules table (Table 8) is then defined. This table shows the set of possible combination between ΔR, ΔL and Δp. Fig. 22 shows some experimental results by using this method implemented on the robot Khepera®.

When the actual environment is either the same as the memorized one (Fig. 22(b)) or not constraining (Fig. 22(c)), the robot navigation is done at high speed by following the planned path. If not (Fig. 22(d)), reactive modules are triggered (from P1 point). The speed is strongly reduced when an obstacle is detected. Then, it increases gradually until the vehicle reaches the sub-goal P2 where the memorized scene is again recognized.

It is possible to verify that the trajectory followed in presence of unknown obstacles (Fig. 22(d)) is very close to the one obtained after including the unknown obstacle in the data base and starting again the planning [15]. In fact the main penalization due to unknown obstacles is the decreasing of the linear speed of the robot.

5. Conclusion

We are interested in the navigation of a mobile robot in partially known environment such as inside an office or a flat. In such cases, a plan of the evolution zone of the robot containing most of its fixed features can be drawn, but numerous undrawn or displaced local obstacles can also been encountered by the robot. So a natural way to obtain an efficient and safe navigation in such an environment is to integrate global planning and local reactive control. The solution we propose here is basically founded on human behavior and mainly implemented through FISs.

The navigation method in an unknown environment is based on the combination of two types of obstacle avoidance behaviors, one for the convex obstacles and one for the concave ones. In the case of convex obstacles, a behavioral agent fusing a “reaching the middle of the collision-free space” behavior achieved by means of STFISs and a goal-seeking behavior, is sufficient. However, the navigation using these strategies can fail if a concave obstacle separates the robot from its goal. In order to solve this problem, a third elementary behavior, of wall-following type, has been developed using another STFIS. Associated to the creation of sub-goals of transition, it permits the robot to skirt round the concave obstacles, before heading again for its goal. The use of FISs to generate elementary behaviors deduced from human being is quite simple and natural. However, one can always fear that the rules deduced from a simple human expertise are more or less sub-optimal. That is why we have tried to obtain these rules automatically. A gradient technique is used which permits the optimization of the output parameters of an FIS through the minimization of a cost function. However, the use of a classical quadratic error as a cost function leads to weight drifting and progressive deterioration of the performances. This problem is solved by a method of weight decay that limits the growing of the weights and allows an efficient on-line learning. Due to the proposed technique, the tedious manual tuning of parameters of an FIS is avoided and the control law is optimized with respect to the actual physical characteristics of the robot.

The a priori knowledge on the environment is memorized and compared to the real scene detected by the robot sensors. If the sensors data in both scenes (memorized and real) are nearly the same, the navigation is done following the planned path at high velocity. If not, it is done under the control of reactive methods. A module, based on fuzzy logic and integrating sensor data, allows going progressively from one of these strategies to the other.

We have used here as a test-bed a real mini robot to prove the effectiveness of the proposed navigation method in spite of very limited calculation resources and a low cost and quite inaccurate sensor system. The implementation of this method on various robots of realistic size for inside works is now in progress and

should be quite easy due to the fact that no explicit model of the robot is needed.

References

[1] M. Benreguieg, H. Maaref, C. Barret, Design of an auto-tuned fuzzy controller: Application to the reactive navigation of a mobile robot, in: Proceedings of the Third IFAC Symposium on Intelligent Components and Instruments for Control Applications, Annecy, June 9–11, 1997, pp. 277–282.

[2] M. Benreguieg, H. Maaref, C. Barret, Navigation of an autonomous mobile robot by coordination of behaviors, in: Proceedings of the Third IFAC Symposium on Intelligent Autonomous Vehicles, Madrid, March 25–27, 1998, pp. 589–594.

[3] C.M. Bishop, Regularization and complexity control in feed-forward neural networks, in: Proceedings of the International Conference on Artificial Neural Networks, ICANN, Paris, Vol. 1, 1995, pp. 141–148.

[4] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation 2 (1) (1986) 14–23.

[5] M. Brunet, Process identification and control by neuro-fuzzy networks, Ph.D. Dissertation, University of Evry, 1996 (in French).

[6] M.Y. Chow, An analysis of weight decay as a methodology of reducing three-layer feed-forward artificial neural network for classification problems, in: Proceedings of the IEEE International Conference on Neural Networks, ICNN'94, Orlando, FL, Vol. 1, 1994, pp. 600–605.

[7] L. Cordewener, D. Meizel, On-line speed monitoring of mobile robots tasks, Engineering Applications of Artificial Intelligence 7 (2) (1994) 151–160.

[8] P. Hoppenot, M. Benreguieg, H. Maaref, E. Colle, C. Barret, Control of a medical aid mobile robot based on fuzzy navigation, in: Proceedings of the IEEE/IMACS International Conference, CESA'96, Robotics and Cybernetics, Lille, July 10–13, 1996, pp. 388–393.

[9] M.I. Jordan, D. Rumelhart, Internal world models and supervised learning, in: Proceedings of the Eighth International Workshop on Machine Learning, Ithaca, NY. 1991, pp. 70–74.

[10] L.N. Kanal, J.F. Lemmer, Uncertainty in Artificial Intelligence, North-Holland, New York, 1988.

[11] O. Khatib, Real time obstacle avoidance for manipulators and mobile robot, International Journal of Robotics Research 5 (1) (1986) 90–99.

[12] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Dordrecht, 1991.

[13] C.C. Lee, Fuzzy logic in control systems: Fuzzy logic controller, Parts I and II, IEEE Transactions on Systems, Man and Cybernetics 20 (2) (1990) 404–435.

[14] H. Maaref, M. Benreguieg, C. Barret, Navigation of a mobile robot in fuzzy tuned artificial potential fields, in: Proceedings of the IEEE/IMACS International Conference, CESA'96, Robotics and Cybernetics, Lille, July 10–13, 1996, pp. 189–191.

[15] H. Maaref, M. Benreguieg, C. Barret, Fuzzy helps to the navigation of an autonomous mobile robot, Journal Européen des Systèmes Automatisés 30 (6) (1996) 839–857 (in French).

[16] E.H. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, Proceedings of the IEEE 121 (12) (1974).

[17] P.J. McVicar, D. Whelan, Fuzzy sets for man–machine interaction, International Journal of Man–Machine Studies 15 (1976) 687–697.

[18] A. Meystel, Autonomous Mobile Robots, World Scientific, Singapore, 1991.

[19] O. Michel, Khepera Simulator, User Manual, Ver. 1.0, 1995.

[20] F. Mondada, E. Franzi, P. Lenne, Mobile robot miniaturization: A tool for investigation in control algorithms, in: Proceedings of the International Symposium on Experimental Robotics, Kyoto, Japan, 1993.

[21] K. Nishimori, S. Hirakawa, H. Tokutaka, Fuzzification of control timing in driving control of a model car, in: Proceedings of the Second IEEE Conference on Fuzzy Systems, San Francisco, 1993, pp. 297–302.

[22] T. Pannerec, M. Oussalah, H. Maaref, C. Barret, Absolute localization of a miniature mobile robot using heterogeneous sensors: comparison between Kalman filter and possibility theory methods, in: Proceedings of the IMACS International Conference on Computational Engineering in Systems Applications, CESA'98, Hammamet, Vol. 4, April 1–4, 1998, pp. 265–267.

[23] F.G. Pin, H. Watanabe, J. Symon, R.S. Pattay, Navigation of mobile robots using a fuzzy behaviorist approach and custom-designed fuzzy inferencing boards, Robotica 12 (1994) 491–503.

[24] J.M. Renders, Biological metaphor applied to process control, Ph.D. Dissertation, Université Libre de Bruxelles, 1994 (in French).

[25] D.E. Rumelhart, G.E. Hilton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA, 1986.

[26] M. Saerens, Connection approach of process control, Ph.D. Dissertation, Université Libre de Bruxelles, 1991 (in French).

[27] A. Saffioti, E.H. Ruspini, K. Konolige, Blending reactivity and goal-directness in a fuzzy controller, in: Proceedings of the Second IEEE Conference on Fuzzy Systems, San Francisco, CA, March 1993, pp. 134–139.

[28] H. Surmann, J. Huser, L. Peters, A fuzzy system for indoor mobile robot navigation, in: Proceedings of the IEEE International Conference on Fuzzy Systems, Yokohama, Japan, Vol. 1, 1995, pp. 83–88.

[29] D.H. Shin, S. Singh, J.J. Lee, Explicit path tracking by autonomous vehicles, Robotica 10 (1992) 539–554.

[30] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

[31] J. Zhang, P. Bohner, A fuzzy control approach for executing sub-goal guided motion of a mobile robot in a partially known environment, in: Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, 1993, pp. 545–550.

H. Maaref received his Ph.D. in 1990. Since 1990, he is Assistant Professor at the University of Evry. In 2000, he received the Habilitation à Diriger des Recherches diploma. He is the Head of the Electrical Engineering Department of the Institute of Technology since 1999. His research interests within the Complex Systems Laboratory of CEMIF concern methods of processing inaccurate and uncertain data with application to autonomous mobile robot and sensorial fusion.

C. Barret was born in Marseille, France, in 1946. He obtained the Aggregation degree in Applied Physics at the Ecole Normale Supérieure de Cachan in 1970 and the Doctorat d'Etat in Electronics at the University of Paris XI, Orsay in 1981. Since 1986, he is Professor at the University of Evry and he was the Head of Electrical Engineering Department of the Institute of Technology from 1986 to 1992. His research interests concern mainly fuzzy control, inaccurate and uncertain data treatment and modeling by learning.