13

A Fuzzy Temporal Rule-Based Approach for the Design of Behaviors in Mobile Robotics

Manuel Mucientes
University of Santiago de Compostela

Roberto Iglesias
University of Santiago de Compostela

Carlos V. Regueiro
University of A Coruña

Alberto Bugarín
University of Santiago de Compostela

Senen Barro
University of Santiago de Compostela

13.1 Introduction II-373
13.2 Fuzzy Control and Fuzzy Temporal Control II-374
Fuzzy Control • Temporal Reasoning: FTR Model
13.3 Description of the Nomad 200 Robot II-378
13.4 Application of FTRs in Wall Following II-380
Introduction • Description of the Control System • Linear Velocity Control • Results
13.5 Application of FTRs in Avoidance of Moving Objects II-391
Introduction • Description of the Problem • Description of the Control System • Moving Obstacle Detection • Obstacle Course Evaluation Module • Behavior Selection Module • Behavior Implementation Module • Analysis and Discussion of Results
13.6 Conclusions II-405
Acknowledgments II-406
References II-406

Abstract. This chapter describes the application of a knowledge and reasoning representation model named Fuzzy Temporal Rules (FTRs) to the field of mobile robotics. This model makes it possible to explicitly incorporate time as a variable, thus allowing the evolution of variables in a temporal reference to be described. Specifically, two behaviors have been implemented on a Nomad 200 robot: wall following and avoidance of moving objects. In both cases it was shown how the use of FTRs not only enables us to determine the tendency of the different variables, but also to filter part of the sensor noise, as the input data are analyzed over temporal references. In this manner a more effective realization of both behaviors is obtained than by conventional fuzzy control.

13.1 Introduction

Currently, one of the principal fields of research in robotics is the development of techniques for the guidance of autonomous robots. Despite the great advances made in robotics in recent years, many complex unsolved problems still exist in this field. The greatest difficulties are due to the nature of the real world (environments difficult to model) and the uncertainty in these environments; knowledge

about an environment is incomplete, uncertain, and approximated because the information supplied by the robot sensors is limited and not totally reliable. Also, the dynamism of the environment in which the robot is located is unpredictable.

There are two clearly differentiated solutions for developing a control architecture: reactive systems[1,2] and deliberative systems.[3] The former operate by sensing the environment and acting (stimulus–response) in a rapid manner, but without the capability for planning, while deliberative systems have to carry out a planning stage before acting. This renders them slow, and the environment may have changed significantly during this period and, thus, actions may already be inappropriate.

Hybrid architectures[4–6] are a half-way point between reactive and deliberative ones; a high-level deliberative strategy is followed when the environment is modeled and the plan to be followed is drawn up. At a low level, the execution layer is reactive, allowing it to act appropriately when faced with changes in the environment. The complexity of the execution layer increases as it must fulfill the different goals proposed by the planner and, at the same time, respond to any changes in the environment. One habitually used strategy to reduce complexity is to break down the execution stage into simple behaviors, such as avoiding obstacles, following a wall, going through a door. Thus, the planner supplies a series of goals that needs to be fulfilled, and different behaviors are activated to do so. The selection of the most suitable behaviors for each situation is implemented by means of an arbitration strategy, while the final order sent to the robot comes from the fusion of the commands of those selected behaviors.

In this chapter we describe two habitual behaviors in the field of mobile robotics (wall following and the avoidance of moving objects) in which a reasoning model called Fuzzy Temporal Rules (FTRs) has been employed. Fuzzy logic has proved to be a useful tool in the field of robotics,[7] as demonstrated in numerous studies carried out for guidance in real environments,[8,9] obstacle avoidance,[10] route planning,[11] and other tasks of interest. This is due, primarily, to fuzzy logic's capability for dealing with imprecise or uncertain measurements, which is of great interest in robotics, where sensorial readings carried out by robots in real environments always have a degree of imprecision associated to them. Despite this, fuzzy logic makes it possible to control a system even when its model is unknown. In this case, the system comprises the robot and the environment surrounding it. Dynamism is a common characteristic of real settings; there are moving obstacles, the environment may be altered by interaction with the robot, people, etc. so that completely modeling the system is an extremely complex task. The use of FTR makes it possible to endow conventional fuzzy control with the tools for carrying out temporal reasoning on system input variables. The significant advantages obtained are twofold: on one hand, it is possible to estimate the tendency of the input variables to examine the evolution of the system; on the other hand, it is possible to filter part of the noise associated to these variables (due fundamentally to erroneous readings of the ultrasound sensors).

The following section introduces fuzzy control, paying special attention to the advantages of fuzzy temporal control, while in Section 13.3 brief mention is made to the robot that is used (Nomad 200). In Section 13.4 wall-following behavior is explained in detail; Section 13.5 deals with behavior for avoiding moving objects; and conclusions are given in Section 13.6.

13.2 Fuzzy Control and Fuzzy Temporal Control

13.2.1 Fuzzy Control

One of the principal advantages of fuzzy control is its ability to incorporate knowledge and schemes of reasoning that are typically human into automation systems. This reasoning incorporates vagueness and imprecision into its description. This capability, inherent to fuzzy logic, for working with vague concepts makes it possible to reproduce experts' knowledge of a domain by means of fuzzy rules,

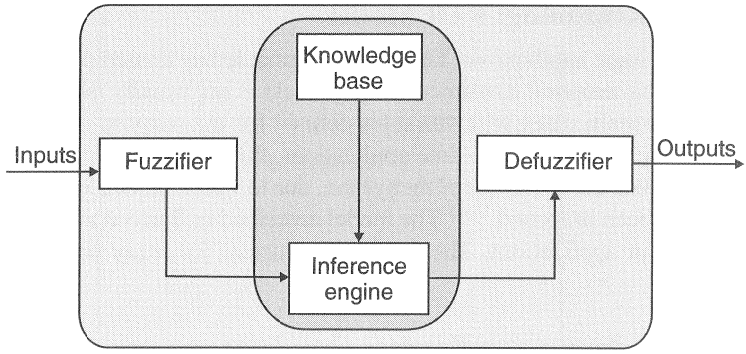

FIGURE II.13.1 Block diagram of a fuzzy control system.

principally when this is expressed by linguistic variables, the values of which cannot be defined precisely (such as, for example, *turn a little*).

Another interesting characteristic of fuzzy control is the lower degree of complexity in the mathematical expressions involved in the description of the reasoning process. This, added to the fact that a fuzzy controller works with a Knowledge Base (KB) comprising a set of more or less simple rules, makes it possible to implement parallel processing, giving rise to carrying out fuzzy inference within a calculation time that, in most of cases, makes its application in real-time problems possible.

Worthy of mention is the greater simplicity in tuning a fuzzy controller (it is relatively simple to verify which rules are being executed at any given moment), apart from the improved robustness due to a high degree of redundancy that allows the system a certain degree of immunity when faced with errors in the KB. Figure II.13.1 shows the typical architecture of a fuzzy controller.

A fuzzy rule base is generally made up of a set of IF–THEN rules that can be expressed as:

$$R^{(l)} : \text{IF } X_1^l \text{ is } A_1^l \text{ and } \ldots \text{ and } X_M^l \text{ is } A_M^l, \text{ THEN } Y^l \text{ is } B^l \tag{II.13.1}$$

where $R^{(l)}, l = 1, 2, \ldots, L$, is the l-th rule, $X_m^l, m = 1, 2, \ldots, M$, and Y^l are linguistic variables of the antecedent and consequent parts, respectively, while A_m^l and B^l are linguistic values (labels) of these variables. One example of a fuzzy rule that follows this outline is

IF velocity is low and the collision time is medium, THEN turn moderately.

In this rule, *velocity, collision time*, and *turn* are linguistic variables, where *low, medium*, and *moderately* are their values, respectively. These fuzzy sets are defined on the universe of discourse U_m that is associated to each variable.

By means of the inference mechanism a mapping of the input fuzzy sets is carried out (those associated to the values described in the antecedent part of the rules). The most widely used inference mechanism in the bibliography is the compositional inference rule,[12] which interprets the process as the Sup-min composition of two relations: the relation induced by the simultaneous observation of the antecedents and the relation induced by the existence of the rules.

In the inference process, each rule is interpreted as a fuzzy implication that establishes a relation between the universes of discourse of the input and output variables:

$$R^{(l)} : A^l \rightarrow B^l \tag{II.13.2}$$

To determine the values of the control variables, we only need to take into account the values of the input variables at the current instant, which gives rise to a series of limitations, such as the impossibility of taking into account the evolution of the variables, or the fact that an erroneous measurement may produce an inadequate control output, as is explained in the following section.

13.2.2 Temporal Reasoning: FTR Model

In the majority of fuzzy logic applications, knowledge is modeled by nontemporal Fuzzy Knowledge Bases (FKBs), where the temporal dynamics of the process is not usually taken into consideration, with the exception of certain cases with variables defined for the purpose ("increase in distance," "accumulated error," etc.). In many real-time applications, this results in a strong restriction on the reasoning possibilities over the dynamics of the system, due to which models explicitly incorporating time as a variable have been proposed.[13–15] The model described in Barro et al.[16] and Bugarín et al.[17] has been adopted for our applications. The following structure for fuzzy temporal propositions is described:

$$X \text{ is } A\langle \text{ in } Q \text{ of } \rangle T \tag{II.13.3}$$

where X is a linguistic variable, A represents a linguistic value of X, T is a temporal reference or entity, and Q is a fuzzy quantifier.

The temporal entities T may represent both fuzzy temporal instants as well as fuzzy temporal intervals. In both cases membership functions are defined on a discrete set of values $\tau = \{\tau_0, \tau_1, \ldots, \tau_k, \ldots, \tau_{\text{now}}\}$, where each τ_k represents a precise temporal instant and τ_0 represents the origin.

We assume that the values of this set are evenly spaced, where $\Delta = \tau_j - \tau_{j-1}$ is the unit of time, whose size or granularity depends on the temporal dynamics of the application with which it is being dealt.

It is considered that all temporal distributions are defined in relation to the temporal point at which the proposition is evaluated, τ_{now} (current instant at each moment, since execution in real time is assumed), as can be seen in the example in Figure II.13.2. When the fuzzy temporal entity T represents an interval, it may require the fulfillment of "X is A" (spatial part of the proposition) for some point of interval T (a situation we refer to as *nonpersistence*), or require its fulfillment throughout the entire interval (*throughout T*: a situation of *persistence*), or that this should be fulfilled for some subinterval (*in the majority of T, in part of T*). An example of the first situation is the proposition "*velocity is high in the last three seconds*," and of the second, *velocity is high throughout the last three seconds*. In the first case, the existential quantifier *in* is included, and it is sufficient for the velocity to have been high at some temporal point which effectively belongs to *the last three seconds* for the compliance of the proposition to be considered high. In the second case (universal quantifier *throughout*), the fulfillment of this must be evaluated on all those temporal points that belong to the support* of *the last three seconds*.

The execution process of a FTR differs from that of a conventional fuzzy rule in the calculation of the degree of fulfillment (DOF), which now not only depends on the current measurement, but also on those corresponding to prior instants. The calculation of DOF is carried out in the model in the following manner: first, the degree of fulfillment of the spatial part of the proposition is calculated, which is defined as the spatial compatibility $sc(\tau_k)$:

$$sC(\tau_k) = \mu_A(X(\tau_k)), \quad \tau_k \in SUPP_T \tag{II.13.4}$$

where μ_A is the spatial membership function associated to the value A of the proposition, and $X(\tau_k)$ is the value observed for the variable X at the temporal point τ_k (whose universe of discourse is U). The spatial compatibilities obtained are modulated by the temporal part of the proposition, so that in

*The support $SUPP_A$ of a membership function μ_A defined on a universe of discourse u is $SUPP_A = \{u \in U/\mu_A(u) > 0\}$.

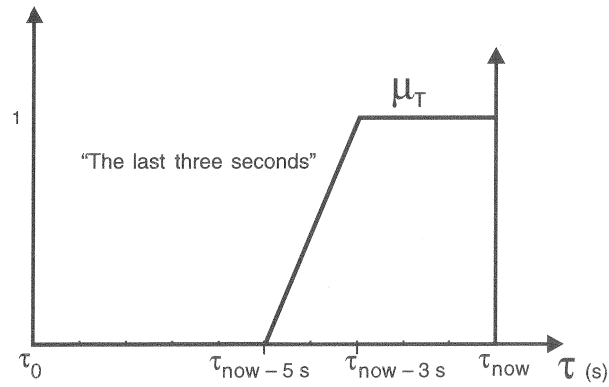

FIGURE II.13.2 Temporal reference μ_T associated with the expression *the last three seconds.*

all the three cases less weight is given to those temporal points of the support of T that do not form part of its core, to thus obtain the degree of fulfillment of the proposition:

- Non persistence: X is A in T

$$DOF = \bigvee_{\tau_k \in \text{SUPP}_T} sc(\tau_k) \wedge \mu_T(\tau_k) \tag{II.13.5}$$

- Persistence: X is A throughout T

$$DOF = \bigwedge_{\tau_k \in SUPP_T} sc(\tau_k) \vee (1 - \mu_T(\tau_k)) \tag{II.13.6}$$

- Intermediate case: X is A in Q of T

$$DOF = \mu_Q \left(\frac{\sum_{\tau_k \in SUPP_T} sc(\tau_k) \wedge \mu_T(\tau_k)}{\sum_{\tau_k \in SUPP_T} \mu_T(\tau_k)} \right) \tag{II.13.7}$$

The operators $\wedge$ and $\vee$ are, respectively, the t-norm minimum and the t-conorm maximum, and μ_Q is the membership function associated to the linguistic quantifier Q. For a given history of values $X(\tau)$ and a linguistic value, depending on the temporal reference T and on the linguistic quantifier employed, it is possible to model different situations enabling the evaluation, in the most appropriate manner, of the different states in which the system to be controlled may find itself and to react to them suitably.

Figure II.13.3 shows an example of the *DOF* calculation for the propositions *velocity is high throughout the last seconds* and *velocity is high in the last seconds*. The process is as follows: first, the spatial compatibility is obtained for each instant τ_k (this spatial compatibility is calculated as the membership degree of $X(\tau_k)$ to the linguistic label high). Thus, five spatial compatibilities are obtained, one for each temporal instant belonging to the support of T. For the persistent type proposition (*throughout*), Equation II.13.6 is applied, which, in the example given, leads to a result of 0.2 (corresponding to $\tau_{\text{now}} - 2\Delta$) while for the nonpersistent type proposition, and applying Equation II.13.5, the final result obtained is 1.0 (corresponding to $\tau_{\text{now}} - 3\Delta$).

Figure II.13.4 shows some definitions for the membership functions μ_Q associated to temporal persistence quantifiers. This rule representation and reasoning model has been used in the two mobile robotics behaviors described in the following sections.

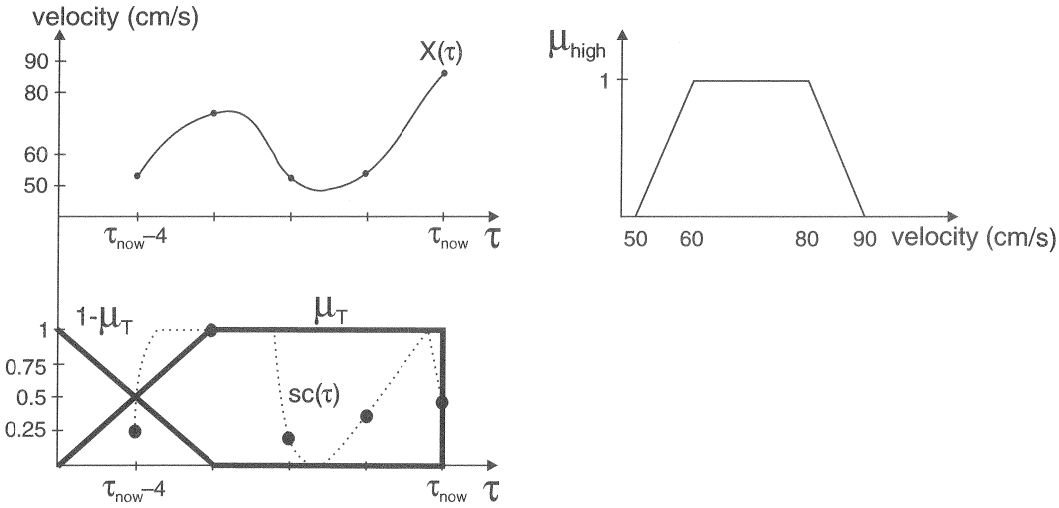

FIGURE II.13.3 Calculation of the DOF for the propositions *velocity is high throughout the last seconds* and *velocity is high in the last seconds*.

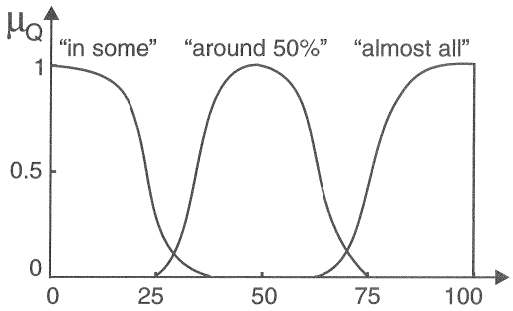

FIGURE II.13.4 Membership functions (μ_Q) of some temporal quantifiers.

13.3 Description of the Nomad 200 Robot

The robot on which the different controllers have been implemented is a Nomad 200 (Figure II.13.5). Table II.13.1 shows some of its principal technical characteristics.

The robot's base has odometric sensors that enable it to know its position, linear velocity, angular velocity, and the turret (upper part of the robot) and base angles. There are also four sensorial systems: bumper, sonar, infrared, and laser. The ultrasound ring (the only sensorial system used in the applications presented in the following sections) comprising 16 sensors supplies the input data requested by the system. The ultrasonic sensors have a range of approximately 6.5 m, and their layout is shown in Figure II.13.6.

To perform its maneuvers the Nomad 200 has a synchronous mechanism for handling its three motors: one for movement, another for turning the wheels, and a motor used to turn the turret independently from its base. It should be pointed out that the robot can turn on the spot, without displacement (null turning radius).

TABLE II.13.1 Characteristics of the Nomad 200

Characteristic	Value
Diameter	53 cm
Height	112 cm
Weight	59 kg
Maximum translation speed	61 cm/s
Maximum linear acceleration	76 cm/s^2
Maximum angular velocity	45°/s

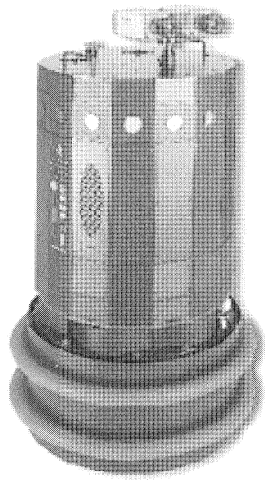

FIGURE II.13.5 Nomad 200 robot (Nomadic Technologies, Inc).

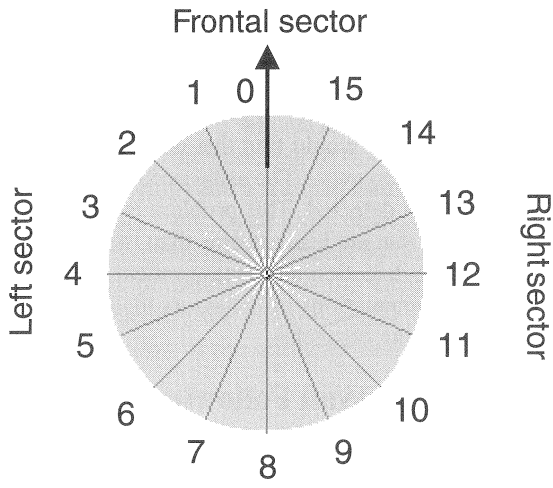

FIGURE II.13.6 Layout of the ultrasound sensors in a Nomad 200 robot. The arrow shows the direction in which the robot advances.

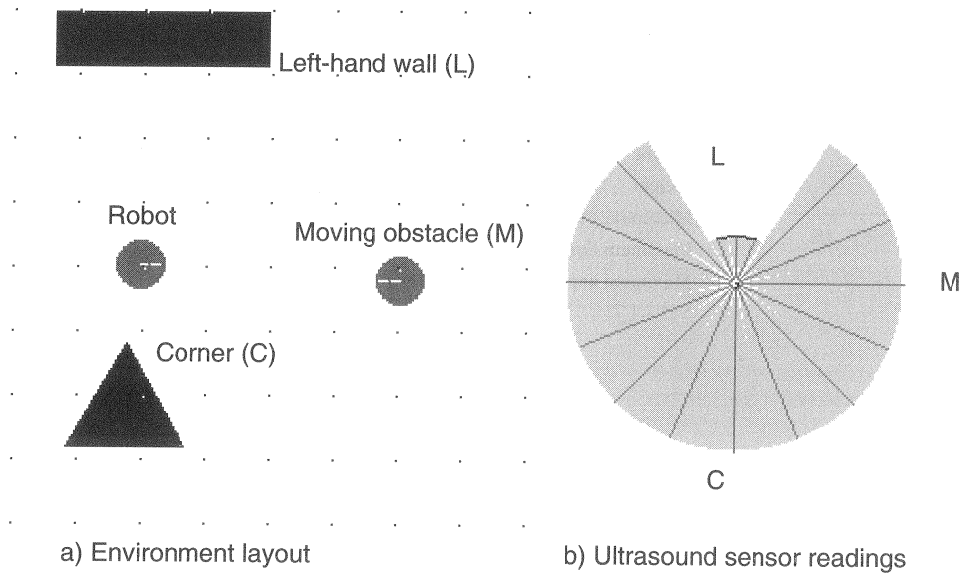

FIGURE II.13.7 Perception failure when the robot approaches an extremely open corner. It can be seen how moving obstacle *M* and corner *C* fail to be detected by the ultrasound sensors.

As is already well known, ultrasound sensors present a series of problems, which may produce a totally erroneous measurement of the distance to an obstacle. Some of these problems are, among others, "crosstalking" which consists of the reception of a pulse by a different sensor from the one that emitted it; specular reflection, which comes about when the incidence angle of the ultrasonic beam with respect to the normal to the wall is high (the smoother the surface of the wall onto which the pulse incides, the lesser the necessary angle). Due to this the reflected energy of the pulse does not return to the emitting sensor, or if it does so, it is after having bounced off other zones, and the measured distance will be erroneous. Last, the width of the beam that ultrasound sensors emit increases proportionally to distance, which leads to errors in the distances measured as well as low angular resolution. Because of this, it will not be possible to detect gaps or to receive information on corners.

Detection problems can be accentuated from the robot turning, above all with concave (closed) corners and, more importantly, in convex (open) corners, as is shown in Figure II.13.7. We can see in this figure how the robot detects the straight wall situated on its left, but the same is not true for the corner, which is undetected. If we opted only to group the sensors into sectors and to select the minimum distance within each sector we would find that no obstacle exists on the right, although the corner is actually located some 50 cm away. The same occurs with the moving obstacle situated in front of the robot, which is also not detected. This problem is compounded in real environments and occurs in a number of situations that are less extreme than the one shown in Figure II.13.7.

The next sections show how these detection and measurement errors can be handled to produce correct control actions for two implementations of behaviors that are usually described in robotics: wall following and avoidance of moving obstacles.

13.4 Application of FTRs in Wall Following

13.4.1 Introduction

One of the most frequently used behavior patterns in robot navigation is wall following.[18–21] This behavior is usually used when the robot needs to move between two given points. Thus, for example, a high-level order may indicate that the robot should move between two rooms in a building. The

module that deals with planning would select the path to be followed and this would be executed by activating behavior patterns such as wall following and door crossing.

In this section we describe a controller for wall-following behavior that supplies the robot with suitable linear and angular velocity values for each given moment. The design has been modularized on the basis of two clearly differentiated objectives: first, to determine in which direction the robot should turn at each instant, so that it remains parallel to, and a certain distance from, the wall it is following; second, the maximum linear velocity at which the robot may move at each moment will need to be determined, without adversely affecting the behavior that is pursued. A good wall-following controller is characterized by three aspects: first, maintaining a suitable distance from the wall being followed; second, moving at a high velocity whenever the layout of the environment permits; and last, avoiding sharp movements, aiming for smooth and progressive turns and changes in velocity.

For all these reasons we have opted for implementing the linear velocity control by means of FTRs,[17] which enable us to collect the knowledge available in a suitable manner, as well as to consider the temporal evolution of the system variables. It is precisely this capacity for contemplating information of a temporal type which, as shall be seen, enables us to filter part of the large amount of sensorial noise and deal with possible erroneous perceptions of the environment due to unsuitable positioning of the robot. Furthermore, more continuous behavior patterns are obtained, which enable the maximum velocities to be selected by taking into consideration the layout of the environment at each instant, which also results in lower navigation time.

This approach differs significantly from others[22,23] in which the value of the control variables is habitually given according to the values of the input variables at a single instant. Other authors[21] have proposed as input to their control systems variables that reflect variations with respect to time derived from the distance and/or the orientation of the robot with respect to the wall, with the aim of anticipating future positions of the robot, although these derivatives only reflect variations between two given instants and not throughout a particular period. On the contrary, when using FTRs we succeed in determining the evolution of a variable throughout a temporal interval, from which it is possible to anticipate future positions of the robot in the environment. Despite the computational requirements being higher when using this scheme, the reasoning model used allows execution of the knowledge base within real-time requirements, which in the case of Nomad 200 are three orders per second.

The other great advantage afforded by the temporal reasoning capability of the FTRs is filtering sensorial noise. A suitable implementation for wall-following behavior requires us to take into account the limitations characteristic of ultrasound sensors, which in some cases render the robot incapable of correctly perceiving the wall that is being followed. A number of strategies for tackling perception problems exist, the most common is the grouping together of the sensors into sectors. Within each sector a single measurement is obtained by means of different solutions: choosing the minimum distance,[22,24] employing the concepts of distributed and general perception,[20,21,25,26] applying a median filter (spatial filtering) and a Kalman filter (the measurements of the current and prior instants are taken into account),[27] etc. In all these cases, the fusion of information originating from sensors is still sensitive to erroneous measurements by one or more of the sensors.

In some studies,[20,25,26] this loss of perception of the environment is avoided by resorting to memorizing the parameters that identify the environmental situation in prior instants to be able to use them for calculating the position of the obstacles at the current instant in the event that the perception is not satisfactory.

13.4.2 Description of the Control System

A graphical description of the velocity control system for wall following is shown in Figure II.13.8. Linear velocity and angular velocity control blocks, respectively, calculate the values for the output

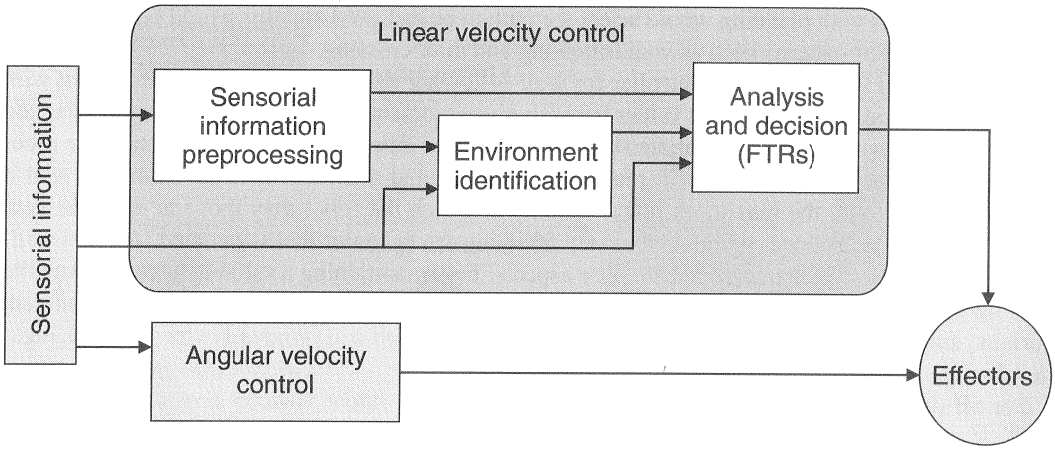

FIGURE II.13.8 Block diagram of the control system.

variables that control the movement of the robot. The calculation of the values of these control variables is carried out independently, in spite of the fact that they are directly linked through the turn radius. We now consider each block in more detail.

13.4.2.1 Angular Velocity Control

The angular velocity controller operates in a very simple manner. In a similar manner to that in Iglesias et al.,[28] information coming from the frontal and lateral sensors is conveniently processed to obtain a binary vector in which each component is associated with a particular direction with respect to the robot. Null components are associated with the presence of possible gaps or free spaces, into which it is possible for the robot to enter. On the contrary, the non-null components indicate which segments of the wall are closer to the robot.

This vector describes with a discrete set of 64 states all the situations in which the robot may find itself. A multilayer perceptron network was implemented for calculating the angular velocity that the robot should attain in each state. As a result in our case, the network is capable of generalizing and generating a suitable action for each one of the possible 64 states.

Other alternatives[22,29] could have been used for the design of this block, but the one described here was chosen due to its simplicity, and because it leads to robust behavior patterns. Moreover, it enables us to easily emphasize the capabilities of the linear velocity controller to guarantee that the robot move with a suitable turn radius, as well as the usefulness and interest of FTRs in the domain of mobile robotics.

13.4.3 Linear Velocity Control

Three modules make up the linear velocity control block (Figure II.13.8): sensorial *information preprocessing, environment identification,* and *analysis and decision.* To facilitate the process of tuning this knowledge and improving the final performance of the system, we consider it appropriate to distinguish between three mutually independent situations, labeled as straight wall, open corner, and closed corner.

Assuming that the robot attempts to follow a wall on its right, we represent any situation in which the robot has to turn to the right, with a low turn radius, and tracing an arc of approximately 90° or higher, as an *open corner* (O_i in Figure II.13.9). If the turn verifies similar conditions, but toward the left, we are in a *closed corner* situation (C in Figure II.13.9); any other situation is placed in the

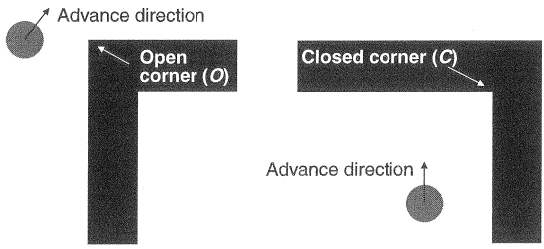

FIGURE II.13.9 Open and closed corner situations.

straight wall category. In the case of following a wall to the left, turns will be produced to the opposite direction.

The knowledge collected in the *analysis and decision* module, as well as the *modus operandi* of the processes included in the *environment identification* module, is based on two different levels of information abstraction: first, the measurements of distance originating from the ultrasound sensors are used. Second, we also use the information coming from the *sensorial information preprocessing module*, which, as will be seen subsequently, is suitable for losing sensitivity or the capability to discern among a number of different environmental configurations, which from a practical point of view can be considered identical.

13.4.3.1 Sensorial Information Preprocessing Module

This module carries out the task of abstracting the sensorial information provided by the robot's ultrasound sensors that perceive the wall being followed or the obstacles close to the robot's forward path. Self-organizing maps[30] are used to process this sensorial information. Instead of implementing one single network in which each neuron processes the information coming from all of the sensors, we use a number of one-dimensional networks, since this results in a superior performance and reduces the complexity of the problem. Each of these one-dimensional networks comprises 15 neurons, and receives measurements coming from three adjacent sensors. Since every measurement is processed by three different networks, nine are required for processing all the information coming from the eleven sensors that are taken into account.

It should be noted that every network learns to classify what the robot "perceives" in one given direction. In general, self-organizing maps can approximate the probability density function of a multidimensional pattern distribution. The network learns to recognize the most frequent patterns. In our case, the learning process was taken into account from a set of sensorial measurements obtained during the movement of the robot. Once the network is trained, its operation always occurs in the same manner; a competitive process takes place at every instant, which produces one winning neuron in each network. This winning neuron will be the one whose learned pattern is the most similar to the scenario the robot perceives in the direction corresponding to that network. It has been noted that for obstacles placed near the robot in one given direction, the winning neuron is usually one of the last neurons in the corresponding network, and vice versa. In this manner, it is possible to calculate a vector $\vec{P}$ in which each component identifies the winning neuron in each of the Kohonen networks. Each component of $\vec{P}$ codifies the global distance to the obstacles detected with a particular orientation to the robot.

13.4.3.2 Environment Identification Module

With this module we are interested in verifying in which of the three previously described situations (straight wall, open corner, closed corner) the robot finds itself at every instant.

To identify these situations whenever the robot is following a more or less straight wall, a reference system (characterized by presenting the y-axis, orientated, as far as is possible, in a parallel manner to the wall) should be constantly updated. In this reference system all points belonging to the wall being followed will be characterized by having the same value x_{ref} for the x-coordinate.

Thus, those sensors perceiving the same wall as the one being followed will be characterized by a coordinate x that is very similar to x_{ref}. In turn, when coordinates associated with the wall being followed stop being received, the vector $\vec{P}$ from the *sensorial preprocessing module* will aid in verifying whether this is due to the presence of an open or closed corner.

13.4.3.3 Analysis and Decision Module

FTRs in the *analysis and decision* module collect knowledge from the task expert in a suitable manner. It is also important for effectively filtering sensorial noise and anticipating future positions of the robot in the environment, as a result of the analysis of the evolution of the system during different temporal intervals. A set of rules describes the correct action to be performed for the three previously mentioned situations. This is described in detail in what follows.

13.4.3.3.1 *Straight Wall Situation*

The rules of the knowledge base for the straight wall situation have been designed to make the robot reach high velocities (up to 40 cm/s) when the distance to the wall of interest, in this case the right-hand one, is suitable (distances in the region of 50 cm are considered to be satisfactory). The robot is parallel to the wall, and furthermore, is closer to the right-hand wall than to the left-hand one. In any other situation the velocity will be reduced appropriately, adapting it to the circumstances in which the robot finds itself.

Assuming the robot follows the wall to the right, the most important variable for selecting the most suitable speed is the distance to the right-hand wall, defined as the minimum distance coming from the sensors in the right sector (Figure II.13.6).

The distance to the obstacles on the left must also influence the calculation of velocity (if the robot is in the center, closer to the right-hand wall, etc.). This information is collected by the variable *distances quotient*:

$$distances\ quotient = distance_{left}/distance_{right} \tag{II.13.8}$$

where $distance_{left}$ is calculated as the minimum distance of the left sector sensors. The distance to the right-hand wall often undergoes variations that do not lead to a change in the linguistic label. To obtain a smoother and more continuous behavior, a good quality controller should foresee and anticipate trend-changing situations. For example, it may happen that during a number of measurements the right-distance label continues to be the same, although the current value for this distance is changing (consistently increasing or decreasing). To also take this information into account, the value of the variable *trend of the right-hand distance* is calculated using the values of the right-hand distance in the current cycle (τ) and in the former one ($\tau - 1$) as:

$$Trend_{right\ distance} = distance_{right}(\tau) - distance_{right}(\tau - 1) \tag{II.13.9}$$

Last, situations exist in which there is no appreciable variation in the distance to the wall of interest, but nevertheless, due to the proximity of the wall, or to the magnitude of the change in velocity, it is necessary to estimate the orientation of the robot with respect to the right-hand wall. For this reason, and using the vector $\vec{P}$ described in the *sensorial information preprocessing* module, a new variable is obtained that evaluates to what degree the robot is facing the wall being followed:

$$Orientation = max\{P_1, P_2, P_3\}/max\{P_4, P_5, P_6\} \tag{II.13.10}$$

This variable is used as a complement to the trend of the right-hand distance, and more specifically when the latter is stable, to better describe the situation in which the robot finds itself. A low quotient indicates that the robot is not facing the wall of interest.

We now show an example of a rule from this knowledge base:

IF the ***distances quotient*** *is high approximately in the last three measurements AND the* ***right-hand distance*** *is medium approximately in the last three measurements AND the* ***right-hand distance*** *diminishes a little approximately in the last two measurements AND the* ***velocity*** *is high, THEN reduce the* ***velocity*** *a little.*

The temporal reasoning ability supplied by the FTRs has been exploited in several propositions from the antecedent part of the rules, as has been seen in the example given. The variable velocity proposition is a fuzzy nontemporal one, given that it is not necessary to carry out a filtering process to eliminate noise, and besides, it is not necessary to analyze the evolution of its values throughout a given temporal interval.

Conversely, the variables obtained from the readings of the ultrasonic sensors (right-hand distance, distances quotient, and trend of the right-hand distance) may indeed contain noise due to the reflections of the ultrasonic beams. Carrying out temporal processing of this information within the temporal reference framework ("the last three measurements," etc.) can avoid spurious erroneous data, which could cause unnecessary alterations in the velocity, enhancing the robustness of the behavior pattern. Furthermore, with the FTRs the evolution of the distance to the right-hand wall can be analyzed to determine its tendency within the temporal interval being analyzed. This means that attention only needs to be paid to those significant and minimally persistent variations of that variable. Thus, we succeed in smoothing out the behavior pattern, as the controller will anticipate future positions of the robot with respect to obstacles, thereby avoiding sharp changes of velocity.

Moreover, the use of FTRs provides flexible representation of knowledge, given that by using different quantifiers it is possible to adjust the percentage of points that need to verify the spatial part of the proposition. Thus, in the rule given as an example, the quantifier *approximately in* means that the degree of fulfillment of the proposition is proportional to the percentage of points in the temporal interval that fulfills the spatial part. In the same manner, the fulfillment of the spatial part could have been required in at least one point (quantifier *in*), at all points (*throughout*), or at a specified percentage of points of the temporal reference.

13.4.3.3.2 *Open Corner Situation*

This situation is defined as one in which the robot has to turn to the right, with a low turning radius, while tracing an arc of almost 90° or more (*O* in Figure II.13.9). This situation is truly complex, as aside from having to apply low velocities to obtain low-turning radius values, it is possible that at some given moments the robot's sensors cease to detect the right-hand wall correctly, even though some of them detect it occasionally (Figure II.13.7). FTRs are suitable tools for dealing with this problem. Even though in some temporal points perception may have been unsatisfactory, analysis of the variables in a temporal interval means that the system's response may be globally correct.

First, we assume that the robot needs to trace a circumference arc that is centered on the vertex of the corner, and with a radius of some 50 cm (the desired distance with respect to the wall). In this way it is possible to estimate what the linear velocity will be at each moment, taking into account the angular velocity at which the robot is moving: $V_{\text{desired}} = V_{\text{angular}} \times \text{radius}_{\text{desired}}$ where $\text{radius}_{\text{desired}} = 50$ cm. Given that V_{desired} is only a value close to the desired one, and is taken as a reference, it is advisable, as a precaution so that the robot does not move too far away from the wall, that the linear velocity at which it moves initially be slightly lower than that one. Nevertheless, as the robot advances, a turning radius lower than the desired one may lead the robot to end up facing or too close to the wall. To avoid this, the velocity should be increased slightly, which will be reinforced by the increased

probability that the open corner will end at any moment. For all these reasons, in the antecedent part of the rules the angle turned by the robot from the beginning of the corner is considered:

$$\text{angle}_{\text{turned}} = \text{angle}_{\text{initial}} - \text{angle}_{\text{current}} \tag{II.13.11}$$

Finally, using the measurements obtained by the sensors in the right-hand section, it is possible to detect the presence of a nearby wall, which could be considered as the end of the open corner. Given that in the final instants it is possible for the robot to have moved with fairly unreliable information, it is necessary to verify that its orientation is suitable. For this reason in this situation, we once again take into account the *orientation* variable, as well as its tendency. These parameters will now be those that predominantly determine which behavior pattern the robot should adopt. Thus, for example, although in the most recent instants the orientation may have been suitable (robot not facing the wall) and the distance to the wall is correct, it will not be advisable to increase the linear velocity if the tendency of the orientation indicates that the robot is facing the wall, since such an increase would take it closer to the wall.

A typical rule from this knowledge base is:

IF the ***right-hand distance*** *is low approximately in the last four measurements AND the* ***left-hand distance*** *is high throughout the last two measurements AND the* ***angle turned*** *is medium or high AND the* ***robot*** *is not facing the wall approximately in the last four measurements AND the* ***orientation*** *is improving approximately in the last four measurements AND the* ***velocity is*** *medium, THEN increase the* ***velocity*** *a little.*

This rule is activated typically in the final instants of an open corner, given that the right-hand wall is close, a fairly or very wide angle has been turned, the robot is not facing the right-hand wall, and furthermore, in the last four measurements the orientation with respect to the wall has improved.

Once again the use of FTRs enables us to carry out filtering of possible erroneous measurements in the distances. This filtering may be more or less demanding depending on the length of the selected temporal interval, and on the linguistic quantifier used (the membership functions μ_Q of the temporal persistence quantifiers used for this application are shown in Figure II.13.10). Thus, for the *right-hand distance*, the use of the quantifier *approximately in* evaluates the percentage of points within the temporal interval that fulfills the spatial part of the proposition, whereas the quantifier *throughout*, used for the left-hand distance, requires the fulfillment of the spatial part of the proposition in all the points of the temporal interval (in this case, this interval only covers two measurements), and as such, is much more restrictive. Finally, the orientation's evolution is evaluated throughout the last four measurements, to verify to what degree this tendency is being maintained.

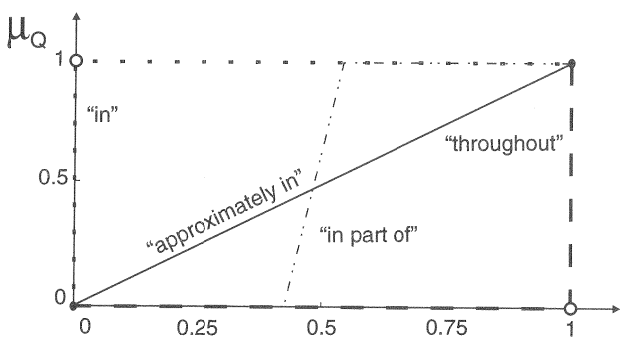

FIGURE II.13.10 Membership functions (μ_Q) of the temporal quantifiers used.

13.4.3.3.3 *Closed Corner Situation*

This situation is characterized by the robot turning to the left with a low turning radius (*C* in Figure II.13.9). As opposed to the other situations, the closed corner is distinguished due to the presence of a frontal object situated at the end of the straight wall that is being followed. For this reason, the frontal distance, calculated as the minimum distance from the frontal sensors, becomes one of the relevant variables that will determine the robot's behavior, even when it is also necessary to take into account the distances to the right and to the left.

To achieve a more precise control action when one of the distances has a low value, its trend will also be analyzed (trend is defined as the difference in distance between two successive cycles) to be able to detect tendencies in the evolution of the values of these variables.

While the robot is still a long way from the corner, its behavior will be no different from that shown in a straight wall situation; however, as it approaches the corner (there will be a reduction in the frontal distance) it has to significantly reduce velocity. Furthermore, the system takes into account that at a closed corner it is possible for the robot to have to turn up to 180°. This will depend on whether there is any obstacle situated to the left of the robot and close to the corner. These different situations can be identified by the fact that once the robot reduces speed and starts to turn, the wall placed on the left will now become a frontal wall. To deal with these situations the evolution of the quotient between the minimum distance to the left and the minimum frontal distance is taken into account, which indicates what type of closed corner the robot is faced with each time. Only when this quotient increases significantly will it be possible to consider that the corner has come to an end.

An example of this situation is found in the following rule:

> *IF the* ***frontal distance*** *is low in part of the last four measurements AND the* ***frontal distance*** *is increasing approximately in the last four measurements AND the* ***quotient between the left-hand and frontal distances*** *is low in part of the last four measurements AND the* ***velocity*** *is low, THEN reduce the* **velocity** *a little.*

It could appear that the robot is starting to detect the straight wall and the closed corner is coming to an end, because *the frontal distance is increasing approximately in the last four measurements*, but given that *the quotient between the left-hand and frontal distances is low in part of the last four measurements*, it is advisable to continue reducing speed, given that there is a wall in the left-frontal zone. This situation is illustrated in Figure II.13.11, where it can be seen how, in spite of the frontal distance in (b) having increased with respect to the situation in (a), the quotient between the left and frontal distances is not high (the distance to the left is not significantly higher than to the front) and hence it is advisable to continue reducing speed slightly to avoid the wall situated in the left-frontal zone. In the case of no wall existing to the left, the quotient between both distances would be high, and thus the robot would start to accelerate, since the closed corner is about to end.

In this closed corner situation various walls are close to each other. It is highly possible, therefore, that due to the turning of the robot rebounding ultrasound signals may give rise to measurements somewhat higher than expected. For this reason temporal analysis of these distances in a wider time frame becomes essential to obtain measurement tendencies that have a high probability of containing noise, and also because in this situation the robot is moving at a notably lower speed. Moreover, the percentage of measurements that must verify the spatial part of the proposition is relaxed, including the quantifier *in part of* which requires the fulfillment of the spatial part in at least 50% of the points in the temporal interval.

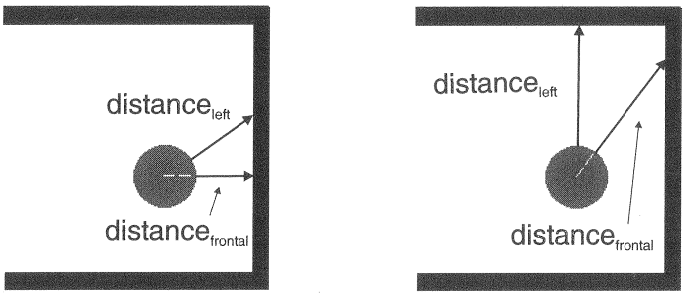

FIGURE II.13.11 Utilization of the quotient between the left and frontal distances.

TABLE II.13.2 Characteristics of the Linear Velocity Control System

	Straight wall	Open corner	Closed corner
Number of rules	108	140	65
Number of antecedents	5	7	8
Number of antecedents in which temporal reasoning is carried out	3	4	7

13.4.4 Results

To verify the system's operation a number of experimental tests have been carried out, in both real and simulated environments, with highly satisfactory results. Two examples implemented on the real robot* are now discussed.

Table II.13.2 shows some of the most relevant characteristics of the linear velocity control system. Despite the large number of rules needed for improving the behavior in any kind of situation, the execution time per cycle of the global control system is 0.22 sec, which allows ample margin for carrying out other tasks. Of the three situations in the environment, the open corner was the one that needed the highest number of rules, owing to its complexity. This is due to the large number of variables needed for its evaluation with all the rules. Conversely, for a closed corner situation, although the number of variables is high, they are not evaluated in all cases, and therefore the number of rules is substantially reduced.

A strict comparison of the results obtained with other control systems described in the bibliography is not possible; however, it is so in a qualitative manner. Thus we can emphasize that unlike Castellano, Attolico, and Distante[22] and García, Mandow, and López-Baldán,[23] our solution takes into account that the linear velocity of the robot may be variable, which enables navigation time to be reduced and velocity adapted to each situation. Furthermore, the results of the tests are given in environments in which the different situations (open corner, closed corner, and straight wall) are connected consecutively, thus increasing the complexity of the environment. The results given in García, Mandow, and López-Baldán[23] and Uribe, Urzelai, and Ezkerra[26] relate to each of these basic situations separately, and it is not possible to evaluate their global performance.

*Video recordings of some tests are available at http://www-gsi.dec.usc.es/areas/robotica_cas.htm.

In a field such as mobile robotics it is not viable to propose a standard "test bank" that would allow us to compare controllers in similar environments, and to consider the different dynamic characteristics of the various types of robots. For this reason, to carry out the results validation, we have chosen a set of examples sufficiently numerous and varied in the situations that have been considered. In this section we describe some of the most interesting cases. In the same manner, we include the values of certain objective parameters that endorse the global operation of the system (average velocity and average distances).

In the first example (Figure II.13.12) a complex, real environment is shown, with closed corners (C_i), open corners (O_i), gaps, doors, etc. To show the complexity of the proposed task, the same figure also shows the readings collected by the ultrasound sensors while the robot was in motion. On the other hand, by means of the robot's trajectory (represented by circles), information on its velocity at each instant (the higher the concentration of marks the lower the velocity) is implicitly included. The first point that should be emphasized is the control system's high degree of reliability and robustness, given that the average distance from the right-hand wall was 49 cm, which falls within the range of satisfactory distances. On the other hand, it can also be seen how the system is capable of satisfactorily resolving all situations, increasing velocity when the robot is following a straight wall (S_6) and reducing it when it approaches an open or closed corner. It is also possible to differentiate the reaction when confronted with an extremely closed corner (C_1) in which an about turn of 180° must be realized, or when it is not necessary to reduce speed so drastically because the is corner not so pronounced (C_3) and a turn of 90° is sufficient. The average velocity in this example was 23 cm/s.

Table II.13.3 shows some of the data measured in the different tests. It can be seen how the average velocity obtained in the example shown in Figure II.13.13 (example 2) is lower than the one obtained in the test shown in Figure II.13.12 (example 1) due to the complexity of the environment in the second example. The maximum and minimum velocities obtained in both environments are identical. It is noteworthy that in both cases, the robot, at some point, has to come to a halt to facilitate the turn, thus average velocities are not higher. With regard to obtained average distances to the right and left, the distances to the right are always close to 50 cm, while the average distances to the left are higher than to the right, and the values obtained in the different environments vary due to their different layouts.

One highly noteworthy aspect is the controller's ability to adapt linear velocity to the situation in which it finds itself, increasing or decreasing it, according to need. This aspect is evaluated in Figure II.13.14, which shows the evolution of velocity as opposed to time in the route shown in Figure II.13.12.

In the diagram, each environmental situation is labeled in the upper section, as well as the average velocity in each section (in cm/s). For straight walls the robot reaches and maintains the maximum

TABLE II.13.3 Data of the Two Examples Described in Figures II.13.11 and II.13.12

	Example 1	Example 2
Average right distance (cm)	49	48
Average left distance (cm)	171	202
Average velocity (cm/s)	23	17
Maximum velocity (cm/s)	44	43
Minimum velocity (cm/s)	0	0
Time spent	2 min 22 s	2 min 53 s

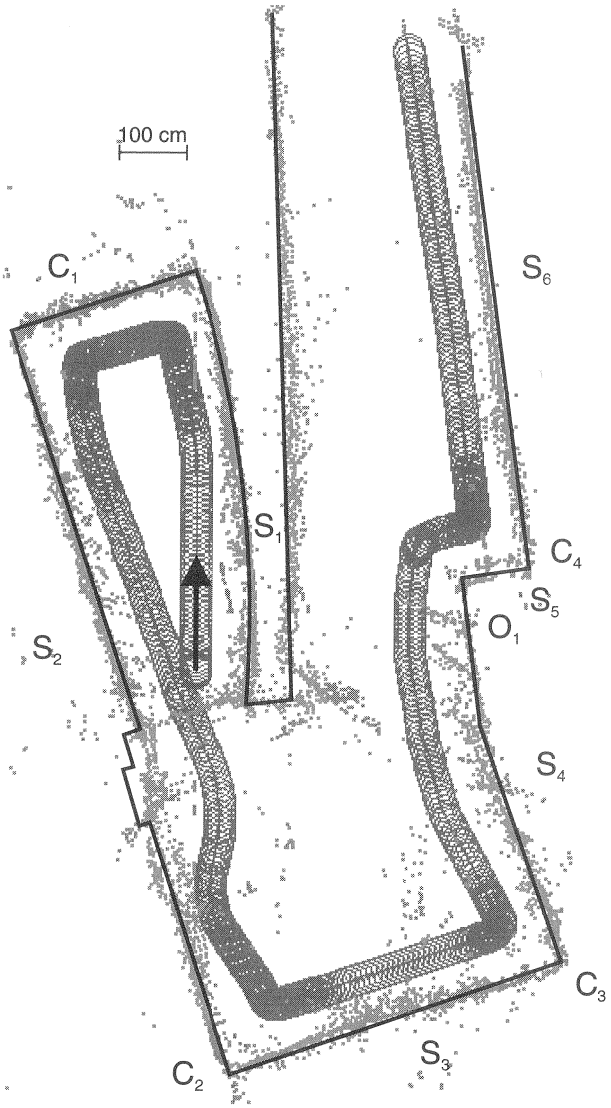

FIGURE II.13.12 Example 1. See text for description.

speed permitted (in the area of 40 cm/s) whenever the wall is sufficiently long (S_1, S_3, S_4, and S_6). In these situations the average velocity is clearly over 30 cm/s. In S_2 and S_5 these average velocities are not attained. In the latter case this is due to the short length of wall S_5. In the case of S_2, when the robot has attained velocities higher than 35 cm/s, available space narrows when going through a door 1.5 m wide. This gives rise to a decrease in velocity to 20 cm/s. Later, velocities over 35 cm/s are reached until, after the door has been crossed, the distance to the right-hand wall is greater than is desired, making it once again necessary to reduce speed. At closed corners the range of velocities is logically much lower than in the straight wall situation (average speeds close to 13 cm/s). In situations such as C_3, where the robot must turn less than 90°, the minimum velocity is 7 cm/s, but in very closed corners (180°) such as C_1 in some cycles the robot must stop to realize the turn, consequently reducing the final average velocity.

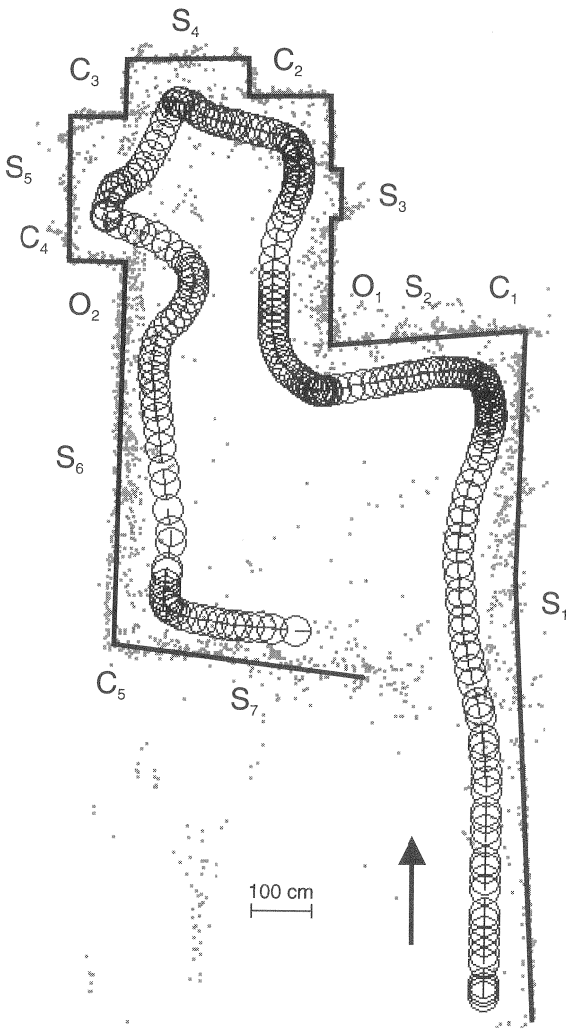

FIGURE II.13.13 Example 2. See text for description.

13.5 Application of FTRs in Avoidance of Moving Objects

13.5.1 Introduction

To the already complicated problem of autonomous robot guidance, we have to add in this case the presence of a moving object approaching the robot. Real-time operation is again one of the characteristics of this problem, since the robot has to be given orders in a rapid manner; if the difference between the time in which the measurements are taken and the moment in which the order is sent to the robot is great, the situation may have changed completely, and the response that was about to be carried out will not be adequate.

There are various approaches for tackling the problem of robot navigation in the presence of a moving obstacle. A number of studies deal with estimating the moving object's future positions. In this manner Elnagar and Gupta[31] use an autoregressive model, placing no restrictions on the trajectory of the obstacle. Chang and Song have[32] also used a neural network to this aim. Spence

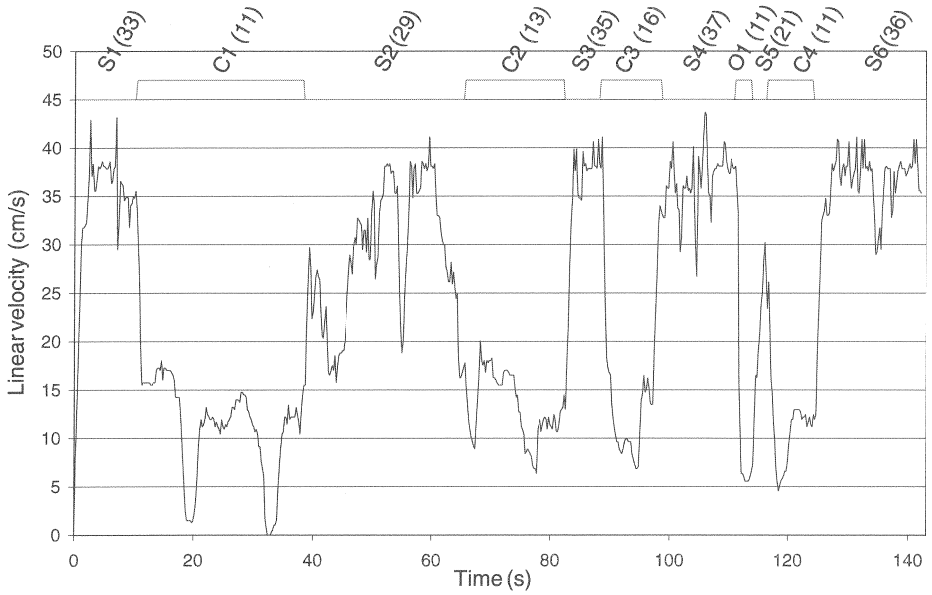

FIGURE II.13.14 Variation of linear velocity along the route shown in Figure II.13.12.

and Hutchinson[33] use a method based on attractive forces, depending on the distance to the goal intended for the robot to reach, and repulsive ones, due not only to the distance between the current state of the robot and the closest configuration in which a collision would occur, but also to the velocity of the moving object. On the other hand, Chakravarthy and Ghose[34] use an approach based on the concept of a collision cone, which allows the detection and avoidance of a collision between a robot and a moving object with an unknown trajectory, there being no constraint with regard to their forms. Gil de Lamadrid and Gini[35] have implemented a system for the monitoring of trajectories to be followed; there is a time limit of the arrival at the goal. The trajectories of the robot as well as of the moving objects are made up of linear segments along which they move at a constant speed. Tsoularis and Kambahmpati[36,37] also describe the avoidance of a moving obstacle, in this case with a uniform and rectilinear movement, resolving it in a geometrical manner. Finally, Garnier and Fraichard[38] and Pratihar, Deb, and Ghosh[39] have developed two fuzzy control systems. As antecedents to the rules, the former uses the velocity of the vehicle to be controlled and the distance between this and the moving object, measured by the different sensors, which also gives its relative position. The output commands are accelerations and changes in turning speed. Another module acts in conjunction with this one, which enables the vehicle to follow a trajectory without straying excessively when avoiding collision with the moving object. On the other hand, Pratihar, Deb, and Ghosh[39] uses the distance of the robot from the moving object and the relative angle that it forms, and the robot acts solely by turning. In both cases the moving objects move in a straight line at a constant velocity.

In all these approaches the robot acts according to the position of the moving object in the immediately preceding instants. In certain cases, this may lead to carrying out precipitated and inadequate actions. Our approach to the problem is to solve this by taking into account the history of more or less recent values of determined variables, which enable us to reflect the different scenarios through which the obstacle has been passing and, thus, verify what its trend is. In this way, one can deduce what the behavior of the robot should be, and take corresponding actions (modification of its speed and/or turning the robot) to obtain a behavior pattern in tune with recent situations. This system is robust as it permits the avoidance of collisions even when the moving object behaves in a

totally unexpected manner. Furthermore, the moving obstacle is allowed to move freely. The need to evaluate past situations and previous values of the variables (which in many cases are fuzzy) and principally, to reason them out, has led us to incorporate FTRs.[40–42]

The representation of knowledge is modularized into three knowledge bases from which the different components of the control system are made independent for greater ease in tuning of the knowledge base, as well as achieving greater simplicity in each one of its components. The system enables the robot to operate with imprecise knowledge and takes into account the physical limitations of the environment in which the robot moves, obtaining satisfactory responses for the different situations analyzed by means of simulation software and in real environments.

13.5.2 Description of the Problem

As has been mentioned, movement of a robot in a dynamic environment is an extraordinarily complex problem. Besides avoiding collision with the moving object, the robot must move in an environment that may have fixed obstacles (walls, etc.) that are restrictions on the movement to be carried out to avoid the moving object. To this, one has to add the restrictions imposed by the robot's characteristics, such as the turn velocity, the linear acceleration, or the range of the sensors.

Unlike other cases, in our approach there is no limitation to the movement of the moving object; it is able to accelerate, brake, turn, etc. When the moving object is detected, the control system cuts in; its objective is for the robot, by means of accelerations and turns, to avoid the moving object approaching it. These robot movements will be limited by the fixed objects present in the environment. The restrictions to the problem are the width of the fixed object-free zone (which we refer to as a passage) and the additional condition imposed that the robot cannot move in reverse.

The aim of the control system is to obtain those control variables that are sent to the robot with each order: angular velocity and linear acceleration. To do this a series of steps is followed: first, the maneuver the moving object is intending to carry out (its trend) is estimated; the control system then selects the robot's behavior for this situation; and finally, the most adequate angular velocity and linear acceleration values are calculated in order to implement this behavior in the optimum manner.

For the input variables of the problem, we have all the parameters of the robot (current position, advance angle, translation velocity, and turn velocity) and the coordinates of the position of the moving object. From these the remaining necessary variables for the solution of the problem can be calculated.

Let $\vec{v}_{robot}$ be the velocity of the robot, $\vec{v}_{obstacle}$ the velocity of the moving object (calculated by simple kinematics based on the positional coordinates in two successive instants τ_l and τ_{l+1}), R_{robot} the radius of the robot, and $R_{obstacle}$ the radius of the moving object (it is assumed that both the robot and the moving object are circular, which does not lead to a loss in generality, see Figure II.13.15a). To be able to determine in a simpler manner the existence or nonexistence of a collision and where it will take place, we carry out a problem transformation,[43] which enables us to go from solving a cinematic problem between two nonpunctual objects to an equivalent static problem. In the equivalent transformed problem the velocity of the moving object is null and its size is $R = R_{robot} + R_{obstacle} + R_{security}$ and the robot is a punctual object with the velocity $\vec{v} = \vec{v}_{robot} - \vec{v}_{obstacle}$. A graphical representation of the transformed problem is shown in Figure II.13.15b.

$R_{security}$ is the minimum distance the robot is permitted to approach the moving object. This is established to maintain a safety margin, which, in any case, avoids a real collision between the object and the robot. Thus the collision test is reduced to verifying the intersection between the straight line given by the velocity of the robot relative to the moving object and the circumference that represents the moving object. In Figure II.13.15b there is an intersection point (collision point), at which a collision will occur. It was assumed in our problem that the robot and the moving object have the

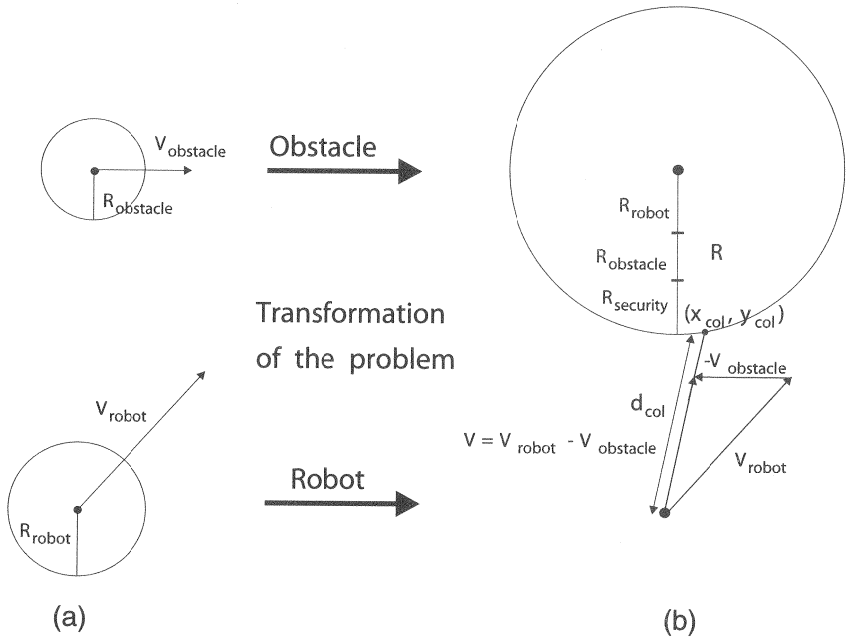

FIGURE II.13.15 Transformation of the original problem (a) in an equivalent one (b), where the robot is a punctual object.

same radius (approximately 25 cm), and the diameter of the robot was taken as the security radius, so that $R = 4 \times R_{robot} (R = 1\,\text{m})$.

Once the equivalent transformed problem has been proposed, a parameter (*noncollision index*) is defined that constantly evaluates the proximity of the current situation with respect to the collision situation.

The *noncollision index* (*nci*) is defined as:

$$nci = \begin{cases} \frac{\sin\alpha}{|\sin\beta|} = \frac{d_c}{R} & \text{if } \alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}] \\ -\frac{d_0}{R} & \text{if } \alpha \in [-\frac{\pi}{2}, -\pi] \\ \frac{d_0}{R} & \text{if } \alpha \in (\frac{\pi}{2}, \pi) \end{cases} \qquad \text{(II.13.12)}$$

where, as can be seen in Figure II.13.16, d_0 is the distance between the robot and the moving object (in the transformed problem the moving object is represented by a circle with radius R; the coordinates of its center are taken as its position coordinates) and d_c is the distance between the moving object and the point with coordinates (x_c, y_c), that is given by the intersection of the straight line in which the robot is moving in the transformed problem (its direction is $\vec{v}$), and the radius of the moving object perpendicular to this straight line. The angle α is formed by the line that joins the robot and the moving object (straight line d_0) with the straight line $\vec{v}$, where:

$$\sin\alpha = \frac{d_c}{d_0} \qquad \text{(II.13.13)}$$

This angle α is measured with respect to the straight line d_0 and increases in a clockwise manner. Furthermore, angle β is the one formed by the straight line that is tangential to the circle with radius R (which represents the moving object in the transformed problem) and the straight line d_0:

$$\sin\beta = \frac{R}{d_0} \qquad \text{(II.13.14)}$$

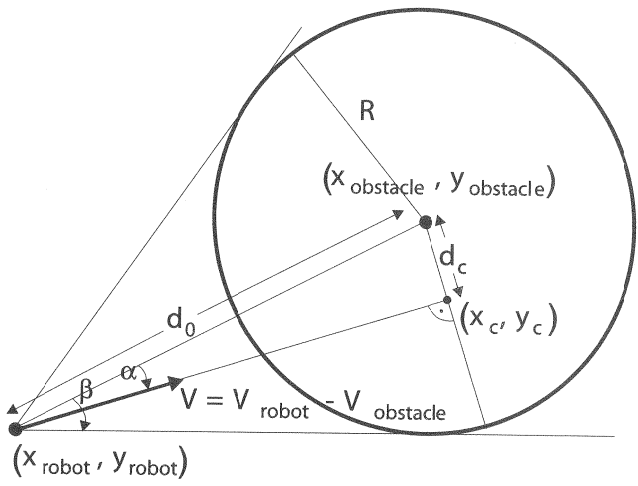

FIGURE II.13.16 Definition of the *noncollision index.*

The *nci* takes values between the interval $[\frac{-d_0}{R}, \frac{d_0}{R}]$, which reduces as the robot approaches the moving object (d_0 decreases in this case). As has been seen in Equation II.13.12, for $\alpha > \pi/2$ and $\alpha < -\pi/2$ the *nci* reaches its maximum and minimum values, respectively. These situations only happen when the robot is moving away from the moving object (in this case, no collision can occur). On the contrary, for values of the *nci* within the interval $[-1, +1]$ a collision situation exists. To obtain these values, angle α must be less than or equal to β (in absolute value), which indicates an intersection between the straight line given by the relative velocity of the robot with respect to the moving object and the circle with radius R. Positive values of the *nci* indicate that the robot is going to collide with the moving object on its left-hand side ($nci \leq 1$), or that it is going to pass before the moving object ($nci > 1$), while negative index values reflect a collision on the right-hand side of the robot ($nci \geq -1$) or that the robot has let the moving object pass ($nci < -1$).

The coordinates of the collision point (x_{col}, y_{col}) are given by the point at which the line $\vec{v}$ intersects with the circle representing the moving object (Figure II.13.15b). d_{col} is the distance separating the robot from this collision point—the point at which the robot will be placed after a certain amount of time (collision time) if the velocities of either the robot or the obstacle remain the same. At this point the robot will be at a distance $R_{security}$ from the moving object.

Variations in the value of *nci* and its temporal evaluation are of great interest for characterizing the dynamic behavior of the obstacle. Thus, for example, the *nci* value may increase because of a decrease in the obstacle's velocity, and it may decrease due to an increase in the obstacle's velocity.*

The *nci* is used as a basis for calculating new parameters related to the evolution of the moving object and/or the robot, since any change in the behavior of either of them will be reflected.

We now describe the intelligent control system in the following section. One aspect that is especially interesting at this point is the previously mentioned necessity to implement temporal reasoning on the evolution of the *nci*. By analyzing the past and present values of this variable, the moving object's current trend can be deduced in an intuitive manner. As an example, if an increase in the *nci* had been produced, but in the last few moments a decrease occurs, it is understood that the previous trend of the moving object to let the robot pass has changed, and has become that of passing

*This is true when the incidence occurs from the left side. For right-side incidences, a change in the *nci* must be made.

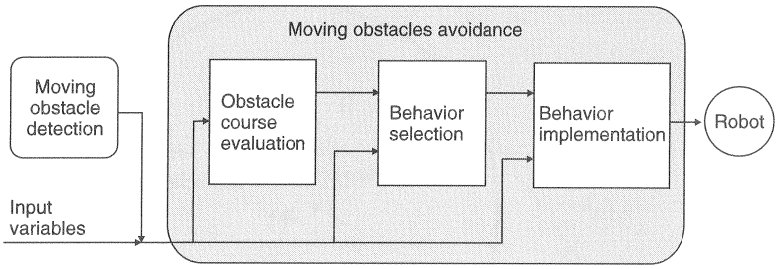

FIGURE II.13.17 Schematic diagram of the control system.

first. In real situations, it will be necessary to distinguish between true changes in the trend and sporadic movements of the obstacle in order to evaluate a situation as changing, a certain persistence or temporal maintenance is required in the new *nci* values. This need to bring temporal intervals into play and to analyze their occurrence in the variables' values has led us to use FTRs.

13.5.3 Description of the Control System

The control system was initially designed bearing in mind that the incidence of the moving object is produced from the left-hand side of the robot. Right-hand side incidences can be dealt with in an identical manner by simply transforming the problem.

Knowledge from the human expert, which has been fed into the system, has been structured into an FKB that has been modularized into three blocks. The objectives of this fragmentation of the FKB are, first, greater ease in tuning the knowledge base. In a nonmodularized knowledge base with a number of rules such as those of this FKB (a total of 117 rules) this process would be extremely complex. Another advantage is that the different blocks have a high degree of independence among themselves, hence modifications in one block do not influence the other blocks.

In order of execution (Figure II.13.17), the modules making up the knowledge base are:

- Obstacle course evaluation module. Its aim is to verify what movement strategy the obstacle is following (if it allows the robot to pass, if it wants to pass, or if it is not aware of the robot).
- Behavior selection module. The aim of this block is to decide on the optimum behavior that the robot should follow in light of the trend of the moving object.
- Behavior implementation module. This final module tries to obtain the angular velocity and linear acceleration with which the robot will most suitably implement the desired behavior for the current situation.

Before describing each of the modules of the FKB, all aspects related to the problem of moving obstacle detection by a real robot are presented.

13.5.4 Moving Obstacle Detection

Detection is based on real-time building of a map of the space occupancy from measurements from the ring of ultrasonic sensors. The degree of occupancy in each cell in the map is calculated by taking into account the number of times it is detected as being occupied and empty (count-based model).[44] This method comprises a very simple sensor model and an accumulation method. Each time a sonar echo is perceived, a value $v_t[i, j] = -1$ is assigned to the cells c_{ij} situated between the transmitter and the possible obstacle, and a value $v_t[i, j] = +1$ to the cells situated on the boundary sector of the sonar cone, independent of its position on it (Figure II.13.18). The accumulation method chosen

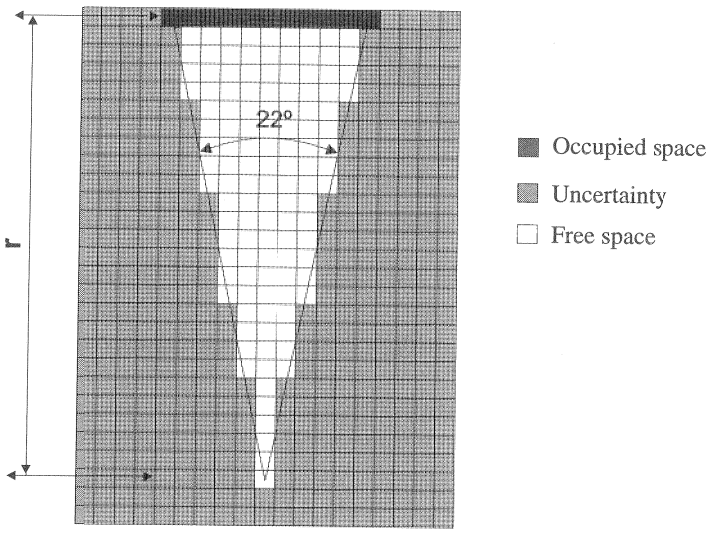

FIGURE II.13.18 Sensor model used, showing the distribution of free and occupied space with regard to one ultrasound measurement.

is the SUM model, in which the occupancy value of each cell $C[i, j]$ of the map is obtained by algebraically adding the different observations:

$$C_{t+1}[i, j] = C_t[i, j] + v_t[i, j] \tag{II.13.15}$$

truncating to a maximum cell occupancy value. The initial value for all cells is 0, i.e., maximum uncertainty.

Detection of a moving obstacle is carried out by using the current measures from ultrasonic sensors and the existing map at the current instant. For all of the measurements provided by sensors, fulfillment of a number of heuristic conditions is requested to assume that a moving obstacle is detected by the sensors. These conditions are based on the following basic idea: a moving obstacle is assumed to be detected in a cell that is currently detected as occupied by a sensor, if that cell was previously detected as empty and is surrounded by empty cells.

In the following sections we examine each of the FKB modules. The description mainly focuses on the most relevant blocks that comprise the control system, rather than on a very detailed description of all the components (variables, processing blocks, etc.) in the controller.

13.5.5 Obstacle Course Evaluation Module

The objective of this module is to estimate the obstacle's movement tendencies, i.e., to attempt to characterize in which dynamic scenario the robot is placed. Evaluating this situation, the robot will assume that the object interfering with its trajectory is trying to pass before it or is letting it pass. In other cases it will not be able to estimate a clear trend in the object's movements.

The input variables for this block are the following: *collision time*, *collision status change*, and *nci trend*. Variable *collision time* estimates the time available before the robot enters into the obstacle's security radius. A low collision time supposes a sharp reaction on the part of the robot to avoid a collision which seems imminent, while a longer collision time enables it to observe the situation and act in a more gradual manner. As has been mentioned, besides indicating whether a collision will occur and what nature it will be, the *nci* reflects any changes in velocity (in module and/or direction) of the robot and the moving object.

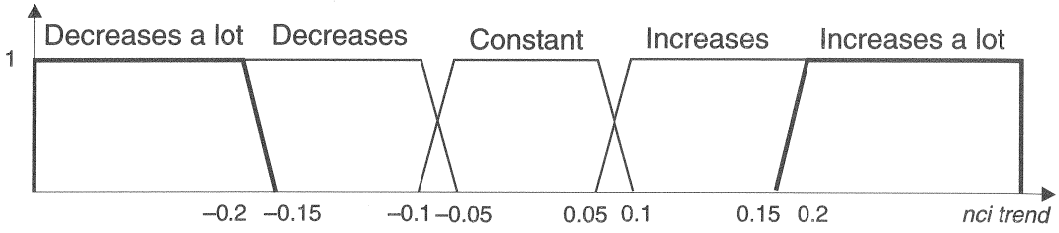

FIGURE II.13.19 Set of linguistic values for variable *nci trend*.

Variable *collision status change* is used to detect situations in which the robot passes from collision in one cycle, to not in collision in a later cycle, or vice versa. Knowing the *nci* values in these two cycles, makes it possible to determine whether the moving object wishes to pass first, or is letting the robot pass. The possible values of the *collision status change* variable considered in this problem are *decrease*, *neutral*, and *increase*.

The *nci trend* variable gathers, on the contrary, a more precise evolution of the trend of the moving object, in which successive differences in the noncollision index are evaluated. To do so, the difference between the values of the index in two successive instants are calculated, and then the mean is found between this difference and the difference in the previous cycle (to attempt to minimize errors due to imprecision in the measurements of the moving object's position):

$$nci\ trend = \frac{nci(\tau) - nci(\tau - 2)}{2} \tag{II.13.16}$$

The values used for this variable are *decreases a lot*, *decreases*, *constant*, *increases*, and *increases a lot* (Figure II.13.19).

The output variable for this block is the trend of the moving object, which estimates the behavior of the obstacle to then be able to act accordingly. In our case the trend is a crisp variable that may take the following values:

- *To give way*. The moving object intends to let the robot pass.
- *Indifferent*. This trend may be due to two reasons: on one hand, the moving object is moving in a random manner (braking, accelerating, or turning without any continuity in its movement) or because the moving object is not varying its speed (module direction).
- *To pass in front*. The moving object is attempting to pass first.

The rules of this knowledge base incorporate temporal reasoning and follow the FTRs model, since a correct evaluation of the object's movement must take previous situations into account, rather than a single summarizing average value. We now analyze the most noteworthy aspects of some representative examples.

One type of rule is:

IF ***collision time*** *is short AND* ***collision status change*** *is decrease in the last two seconds AND* ***nci trend*** *is not increasing throughout the last second, THEN* ***obstacle aim*** *is to pass in front.*

The meaning of this rule is that in a situation of relative proximity between the obstacle and the robot (*collision time is short*), it is assumed that the trend of the former is to pass in front if there has recently been at some point a decrease in the collision status change (this may occur when, for example, a noncollision situation with a high *nci* value changes into a collision situation – *collision status change decreases in the last two seconds*) and furthermore, even more recently, the *nci* has maintained its value or decreased (*nci trend is not increasing throughout the last second*). A strict decrease in the *nci* is not required, as this decrease has been produced implicitly since a change in the *collision status* has occurred.

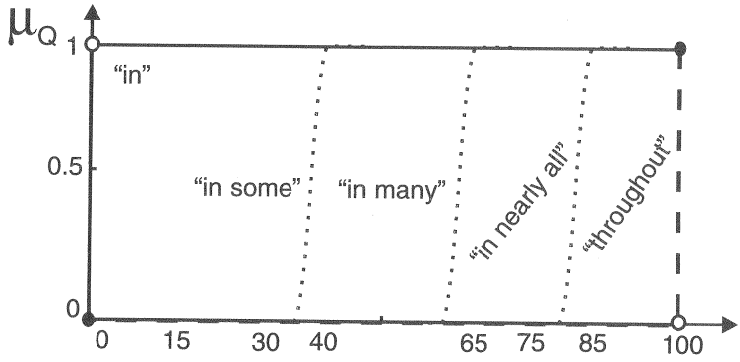

FIGURE II.13.20 Membership functions (μ_Q) of some of the temporal quantifiers used.

In general, if *collision status change* decreases, and the *nci* decreases or remains constant, the trend will indicate that the moving object intends to pass (for a *to give way* trend an increase in the *nci* is required), while if subsequent to the *collision status change* decrease the *nci* increases, the trend will be *indifferent.*

Another possible situation is no *collision status change.* In this case the trend will be *to give way* if the *nci* increases substantially (for a *to pass in front* trend a significant decrease in the *nci* would be required):

IF ***collision time*** *is medium AND* ***collision status change*** *is neutral in the last three seconds AND* ***nci trend*** *is increasing a lot in the last three seconds, THEN* ***obstacle aim*** *is to give way.*

This rule analyzes whether for medium collision time a *collision status change* has not occurred at any point of the interval *the last three seconds.* In this case, a *to give way* trend is assigned to the moving object, if the *nci* has increased significantly at some point of this interval.

This increase may be given, for example, by a decrease in the moving object's velocity module. However, this decrease may take a more gradual form and have almost the same final effect (braking is not so sharp, thus the moving object will be closer to the robot). Rules like the one that follows have been introduced into the knowledge base to resolve this type of situation:

IF ***collision time*** *is medium AND* ***collision status change*** *is neutral in the last three seconds AND* ***nci trend*** *is increasing in at least a few points of the last three seconds, THEN* ***obstacle aim*** *is to give way.*

The difference in this case is that the requirement for the increase in the *nci* is not as strict (the only requirement in this rule is that the nci should increase) so that the change in the index may be lower, but in this rule the increase is needed *in at least a few points of* the interval *the last three seconds* (*a few* represents approximately 30%). Definitions of some of the temporal quantifiers used for this application are shown in Figure II.13.20.

13.5.6 Behavior Selection Module

The objective of this block is to determine the type of behavior that should be adopted by a robot faced with the trend given by the moving obstacle's current situation. The robot implements an ample set of behavior types to try to resolve the multiple situations in which it may find itself. The input variables for this module are *collision time, trend,* and *limit situation.*

Limit situation is a crisp variable that indicates when the robot is in an extreme situation in which it will attempt to leave the trajectory of the moving object as quickly as possible. The conditions for

the robot to be in a *limit situation* are two: it should be placed in the trajectory of the moving object,* and the incidence of the moving object should be *extremely frontal* or *extremely rear*.

As has been mentioned, a good number of possible behavior patterns exist. Moreover, given two equal behavior patterns, they do not necessarily have to be implemented in the same manner, rather this realization of behavior will depend on a series of variables; this is the task of the final block (behavior implementation module). The types of behavior that exist and the general descriptions of their implementations are:

- *To give way*. In this behavior pattern the robot lets the moving object pass by, and it does so braking and, sometimes, also turning.
- *To observe*. In this situation, the robot maintains its velocity (module and direction). This is normally because the trend of the moving object is not clear.
- *To pass in front*. Here, the robot attempts to pass before the moving object (for example, by turning and accelerating).

In addition to these basic behavior patterns, there is a specific behavior for *limit situations*. In these situations the trajectories of the robot and of the moving object are fairly parallel and it is essential for the robot to turn to move out of the path of the moving object. The behavior patterns for *limit situations* are characterized by trying to leave the path of the moving object as quickly as possible. To do so the robot will need to accelerate (as braking will not take it out of the moving object's path) and turn sharply (the turn is made in the direction with the greatest margin).

The most common rule used for the determination of the behavior is:

IF ***collision time*** *is high AND the* ***obstacle's trend*** *is to give way, THEN the* ***robot's behavior*** *is to pass in front.*

In this rule behavior heavily depends on the trend variable. If the trend is *to give way*, as a general norm, the behavior will be *to pass in front*, while if the trend is *to pass in front* the behavior will be *to give way*. For an *indifferent* trend the selected behavior will take into account the collision time. For high collision times the robot will act aggressively, and hence the corresponding behavior will be *to pass in front* while for low collision times the robot will act in a more conservative manner, implementing a *to give way* behavior. With medium collision times, the robot will adopt intermediate tactics, and the behavior pattern will be *observe*, due to which the robot will wait for future changes in the trend of the moving object.

In certain cases, the behavior patterns obtained in this block may be modified. This happens when the selected behavior cannot avoid collision (for example, the robot cannot turn, or it is in the trajectory of the obstacle and cannot brake, etc.). A set of criteria was implemented for these cases and decide the most favorable behavior pattern for the existing conditions.

13.5.7 Behavior Implementation Module

The aim of this block is to obtain, from a series of input data (*trajectory*, *behavior*, *collision time*, *robot's velocity*, *deviation*, and *incidence*) the angular velocity and the linear acceleration commands that will be sent to the robot. The crisp variable *trajectory* indicates whether the robot is in the path of the moving object or not.

---

*The trajectory of the moving obstacle is represented by a band, the width of which is equal to the diameter of the mobile object in the transformed problem.

Since the robot moves in the direction of a goal, any turn that makes the robot move toward the goal is labeled as favorable, while a turn that takes it away from this path is unfavorable. The variable *deviation* is defined for measuring how favorable a turn in a determined direction is. It may take the following values: *negative*, *null*, and *positive*. Negative deviations describe very favorable turns (movements toward the goal); null values will also allow the turn to be made, although this may produce a movement slightly away from the path. Positive deviations describe turns that will only be implemented in situations in which they are imperative for avoiding a collision.

The positive or negative sign of *deviation* is selected taking into account the following: if the robot is moving toward a point situated to the left of the goal, any turn to the right will be considered favorable since it will take it toward the goal, while a turn to the left will be considered unfavorable because it will take it away from the goal.

The behavior *to pass in front* will usually imply a turn to the right.* If it happens that the trajectory of the robot is placed to the right of the goal, the requested turning to the right will not be favorable (positive *deviation*). For the same behavior, if the trajectory of the robot is placed to the left of the goal, turning to the right is considered favorable and thus *deviation* is negative.

For *to give way* behavior, the turn is usually produced toward the left, so a turn in this direction is favorable (negative sign for the *deviation*) when the robot moves toward the right of the goal and unfavorable in the contrary case.

Variable *incidence* indicates whether the incidence of the obstacle is frontal, transversal, or from the rear. It is necessary to distinguish among these situations, as the optimal manner for avoiding collision is different. The *incidence* is defined as:

$$incidence = \frac{\pi/3 - angle}{\pi/3} \tag{II.13.17}$$

where $angle \in [0, \pi]$ is the angle formed between the velocity of the robot ($\vec{v}_{robot}$) and the relative velocity between the robot and the moving object ($\vec{v}$). The set of values for the variable is made up of: *rear*, *transversal*, *frontal*, *extremely frontal*, and *extremely rear*.

The 72 rules making up the FKB of this module enable us to implement different behavior patterns for each of the situations in a precise and adequate manner. The rules in this block can be grouped according to the behavior to be implemented and the type of incidence.

We now analyze a rule that represents those in which the behavior is *to pass in front* and the incidence is transversal. These rules are generally characterized by implementations of behavior that consist of right turns and accelerations. Thus, for example:

IF the ***robot's behavior*** *is to pass in front AND the* ***collision time*** *is medium AND the* ***robot's velocity*** *is medium AND the* ***deviation*** *is null AND the* ***incidence*** *is transversal, THEN increase* ***velocity*** *quite a lot AND* ***turn*** *a little.*

For high collision times, the reactions should be gentle (slight turns and accelerations) while for low collision times the system usually applies maximum turn and acceleration to avoid collision. In general, the aim is to avoid turns (to not move away from the trajectory the robot was following) except when these are favorable (negative deviations). For low collision times, this is not fulfilled (there is no other solution for avoiding the collision), and for positive deviations there will be a turn, although less intense than those implemented for negative or null deviations.

*Recall we are assuming that the incidence is from the left side.

TABLE II.13.4 Characteristics of the Tests Performed

Maximum Translation Velocity	61 cm/s
Range of distances between the robot and the obstacle	0.5 to 6 m
Range of collision times	1 to 30 s
Range of angles of incidence between the robot and the obstacle	0 to 2π
Types of movements of the obstacle	Accelerations, decelerations, and turns
Types of trajectories of the obstacle	Straight line, curved line, zigzag (and their combinations)

13.5.8 Analysis and Discussion of Results

Since the entire control system has been presented, we now analyze some representative examples of the robot's avoidance of a moving object. The system has been tested for a number of cases, simulated and real, to verify its validity and effectiveness. These tests have included the whole range of possible velocities, for both the robot and the moving object, as well as different angles of incidence (frontal, rear, left transversal, right transversal, etc.) as shown in Table II.13.4.

Furthermore, with the objective of making the simulations as realistic as possible, the tests were carried out with randomly introduced noise in the position of the moving object, to simulate the imprecision of the robot's real ultrasound sensors. This noise is a function of the distance between the robot and the moving object (the greater the distance, the higher the noise) and tests were carried out for a maximum 10% error.

Avoidance of the moving object was also made more difficult for the robot. To do so, when the robot started to implement a behavior pattern to avoid the collision, the moving object reacted by attempting to provoke a new collision situation.

Simulations have also been carried out in which the moving obstacle has a complex trajectory (for example, moving in a zigzag manner that takes up the entire passageway), or effecting a series of turns. The response was satisfactory in all cases, which demonstrates the robustness of the system as well as its reaction capabilities.

The examples we have chosen to illustrate the robot's behavior are given with graphical representations in which the trajectories of the moving object and the robot are described. Those of the former were chosen to show a selection of changes in the module and the direction of the velocity that face the robot with varied scenarios.

13.5.8.1 Example 1

In this first example, a moving object frontally approaches the robot, both moving at a velocity of 25 cm/s, in a situation in which at the beginning, there is not going to be a collision. Thus the robot maintains its speed in terms of module and direction (Figure II.13.21). When the moving object approaches to a distance of 380 cm from the robot, it starts to turn toward its left (Point A, Figure II.13.21) with an angular velocity of 30°/s, moving onto a collision course, and the control system starts to act. As a result of this turn, the *obstacle course evaluation* module correctly interprets that the moving object wishes to pass (the moving object has made a turn to its left). As the robot is now in the path of the moving object and the incidence is frontal, the behavior selected will be *limit situation*.

Under these circumstances the only possibility of avoiding collision is turning to the right. This behavior (*limit situation*) is characterized by high acceleration and turn, which will be more intense the lower the collision time is. In this situation the collision time is medium, and once the command has been sent to the robot, in the following cycle the robot is still on collision course (acceleration and turn were insufficient). The process is repeated, but this time with the collision time being

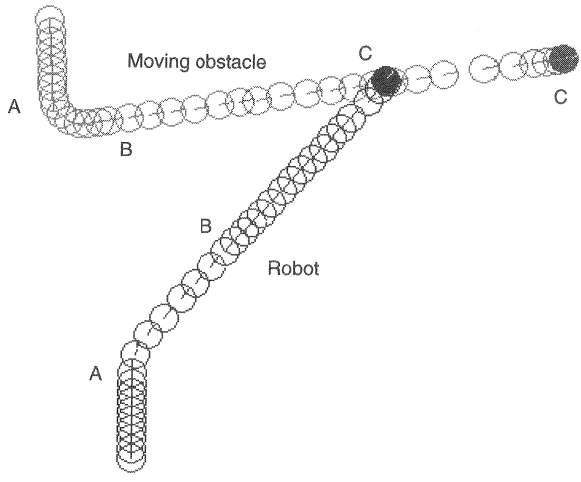

FIGURE II.13.21 Example 1. A, B, and C are the positions that the robot and the obstacle occupy in three time instants of the time interval represented.

medium-low, so that finally in the next cycle (with collision time already low), the robot succeeds in leaving the collision situation.

Once the collision has been avoided, the moving object continues turning until it reaches 45°, while the robot maintains a velocity of 61 cm/s. When the moving object reaches point *B*, it increases its velocity from 25 to 61 cm/s and once again the control system has to act because of the collision situation. Due to this acceleration, the *obstacle course evaluation* module indicates that the moving object is attempting to pass first, and the behavior selected is *to give way*. Its implementation for low collision time consists of braking sharply, taking the robot from a velocity of 61 cm/s to one of 36 cm/s, thus avoiding impact.

13.5.8.2 Example 2

In this second example a situation is described in which the moving object starts by letting the robot pass then changes strategy and passes first. The robot responds to these situations by simply varying its velocity (Figure II.13.22). At the start, the robot detects the moving object, and since it has no data stored (the moving object has entered within range of the robot's sensors for the first time), the tendency selected is *indifferent*. Because the collision time is high and the tendency of the moving object is *indifferent*, the robot adopts an aggressive behavior and attempts to pass first. The implementation of this behavior, for null *deviation* and medium robot velocity (25 cm/s), is a small acceleration. Due to this acceleration (which increases the robot's velocity from 25 to 33 cm/s) and because the moving object has started to turn to the right (point *A*), the *obstacle course evaluation* module now detects that the *nci* has increased substantially and the tendency will thus be *to give way*. The behavior of a robot faced with this attitude on the part of the moving object is to try to pass first, and it continues with the strategy begun in the previous cycle. Once again this behavior is implemented with a small increase in the robot's velocity, up to 38 cm/s, avoiding the collision.

A number of cycles later, the moving object changes its behavior by turning to its left and increasing velocity from 25 to 61 cm/s (*B*), which leads to a sharp decrease in the *nci*. Under these circumstances, the intention of the moving object is clearly to *pass first*. The *behavior selection* module selects *to give way*, and since the collision time is low, it acts by applying the maximum deceleration possible, decreasing the robot's velocity to 13 cm/s, thus avoiding collision.

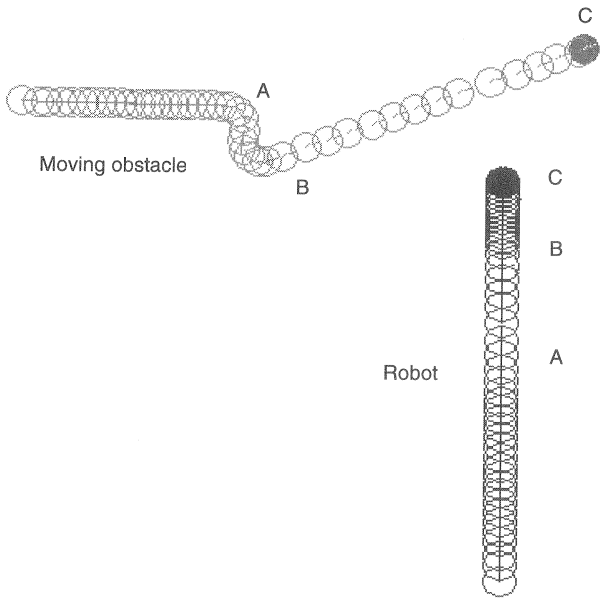

FIGURE II.13.22 Example 2. A, B, and C are the positions that the robot and the obstacle occupy in three time instants of the time interval represented.

13.5.8.3 Avoidance in Real Environments

In real environments, using people as moving obstacles, different tests have been carried out showing that the system is robust and reliable, despite the low precision the method for ultrasonic-based moving obstacle detection produces. A real example of collision avoidance* is shown in Figure II.13.23. Ultrasonic sensor measurements are represented by points, possible moving obstacles by (×), and detected moving obstacles by (+). The solid line groups all detected moving obstacles that correspond to the same existing obstacle. In general, all possible obstacles correspond to detected ones (represented by (*), as happens for $M1$), although the final position may be calculated as the arithmetic mean of two detected positions.

The arrow on the right side of Figure II.13.23 represents an approximation of the real trace of the moving obstacle. At the beginning, the robot moves at a constant velocity (18 cm/s) and detects the moving obstacle (this happens when the obstacle is placed at $M1$ and the robot at $R1$), but takes no action because the trend of the obstacle is indifferent and the collision time is medium, so the robot will observe. Later, when the obstacle is placed at $M3$–$M4$, the *obstacle course evaluation* module detects that, because the obstacle turns to the left, its tendency is *to give way*. This is done by means of the following rule:

IF ***collision time*** *is medium AND* ***collision status change*** *is increase in the last three seconds AND* ***nci trend*** *is not decreasing throughout the last two seconds, THEN* ***obstacle aim*** *is to give way.*

Due to this tendency, selected behavior is *to pass in front* and it is implemented by means of acceleration and turning to the left (since the obstacle's incidence is from right to left). This behavior is repeated for the next two operation cycles, until collision is avoided. To reach this, the robot

*A video recording of a similar test is available at http.//www-gsi.dec.usc.es/areas/robotica_cas.htm.

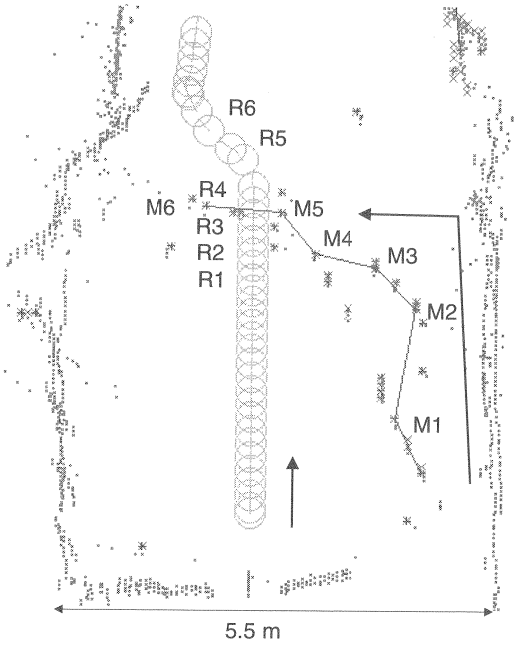

FIGURE II.13.23 Example of collision avoidance with a person who lets the robot pass.

needs to increase speed up to 45 cm/s and turn 40°. These values are reached two iterations after the first collision situation was detected, because robot's motor response is not instantaneous. One of the greatest difficulties encountered is that the obstacle's real position is often unknown, so the robot must use the last detected position. Thus, when the obstacle is placed at $M5$, another collision situation occurs after the change of obstacle position (1 m). This situation is solved by increasing speed and turning to the right, avoiding collision two cycles later. From final positions $R6$ (for the robot) and $M6$ (for obstacle), the robot resumes its trajectory toward the goal point, since the moving obstacle avoidance system does not take control anymore.

13.6 Conclusions

In this work we have described the application of fuzzy temporal controllers in the field of mobile robotics. Fuzzy logic is a highly useful tool in robotics, not only due to its reasoning capabilities with imprecise and/or uncertain values, but also for those situations in which the system model to be controlled is highly complex, or cannot be obtained.

In addition, FTRs exhibit the ability to carry out temporal reasoning by analyzing the evolution of variables throughout temporal references. In this manner, we succeed in determining the tendencies of variables and in filtering sensorial inputs. Hence, the control command not only depends on the values of the input variables at the current temporal point, but also takes into consideration previous values that may be highly significant in given situations.

The usefulness of this knowledge representation and reasoning model (FTRs) has been demonstrated with the implementation of two behaviors on a Nomad 200 robot: wall following and avoidance of moving objects. In both cases it has been shown, by means of tests in real environments, how due to their greater expressive capability the use of FTRs result in an increase in the reliability and robustness of these behaviors as well as their richness.

Acknowledgments

The authors wish to acknowledge the support from the Secretaría Xeral de I + D of the Xunta de Galicia through grant PGIDT99PXI20603A and from the European Commission and Spanish CICYT through grant 1FD97-0183.

References

1. Brooks, R.A., A robusted layered control system for a mobile robot, *IEEE Journal on Robotics and Automation*, RA-2, 24, 1986.
2. Arkin, R.C., Motor schema-based mobile robot navigation, *The International Journal of Robotics Research*, 8, 92, 1989.
3. Meystel, A., Autonomous mobile robots: Vehicles with cognitive control, vol. 1 of *Automation*, World Scientific, Singapore, 1991.
4. Konolige, K. et al., The Saphira architecture: A design for autonomy, *Journal of Experimental and Theoretical Artificial Intelligence*, 9, 215, 1997.
5. Gat, E., Three-layer architectures, in *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, Kortenkamp, D., Bonasso, R.P., and Murphy, R., Eds., 195, AAAI Press/The MIT Press, Cambridge, 1998.
6. Arkin, R.C., Integrating behavioral, perceptual, and world knowledge in reactive navigation, *Robotics and Autonomous Systems*, 6, 105, 1990.
7. Saffiotti, A., The uses of fuzzy logic in autonomous robot navigation, *Soft Computing*, 1, 180, 1997.
8. Lee, E.T., Applying fuzzy logic to robot navigation, *Kybernetes*, 24, 38, 1995.
9. Toda, M. et al., Navigation method for a mobile robot via sonar-based crop row mapping and fuzzy logic control, *Journal of Agricultural Engineering Research*, 72, 299, 1999.
10. Althoefer, K. et al., Fuzzy navigation for robotic manipulators, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6, 179, 1998.
11. de Lope, J., Maravall, D., and Zato, J.G., Topological modeling with fuzzy petri nets for autonomous mobile robots, in *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-98-AIE)*, no. 1416 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1998, 290.
12. Zadeh, L.A., Outline of a new approach to the analysis of complex systems and decision-making approach, *IEEE Transactions on Systems, Man and Cybernetics*, 3, 28, 1973.
13. Chen, Z., Fuzzy temporal reasoning for process supervision, *Expert Systems*, 12, 123, 1995.
14. Más, O., Palá, P., and Miró, J.M., Razonamiento aproximado en presencia de conceptos temporales difusos, in *Actas del VI Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF '96)*, Oviedo (Spain), 1996, 185.
15. Qian, D.Q., Representation and use of imprecise temporal knowledge in dynamic systems, *Fuzzy Sets and Systems*, 50, 59, 1992.
16. Barro, S. et al., Petri nets for fuzzy reasoning on dynamic systems, in *Proceedings of the 7th IFSA World Congress*, Prague, 1997, 279.
17. Bugarín, A. et al., Reasoning with Fuzzy Temporal Rules on Petri Nets, *Fuzziness in Petri Nets*, vol. 22 of *Studies in Fuzziness and Soft Computing*, Physica-Verlag, Heidelberg, 1999, 174.
18. Mucientes, M. et al., Control de la velocidad de un robot móvil mediante reglas temporales borrosas, in *Actas del X Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2000)*, Sevilla (Spain), 2000, 127.
19. Mucientes, M. et al., A fuzzy temporal rule-based velocity controller for mobile robotics, *Fuzzy Sets and Systems* (special issue on fuzzy set techniques for intelligent robotic systems), accepted.

20. Arrúe, B.C. et al., Fuzzy behaviours combination to control a non-holonomic robot using virtual perception memory, in *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (Fuzz-IEEE '97)*, Barcelona (Spain), 1997, 1239.
21. Braunstingl, R., Mujika, J., and Uribe, J.P., A wall following robot with a fuzzy logic controller optimized by a genetic algorithm, in *Proceedings of the International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium*, 5, Yokohama (Japan), 1995, 77.
22. Castellano, G., Attolico, G., and Distante, A., Automatic generation of fuzzy rules for reactive robot controllers, *Robotics and Autonomous Systems*, 22, 133, 1997.
23. García-Cerezo, A., Mandow, A., and López-Baldán, M.J., Fuzzy modelling operator navigation behaviours, in *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (Fuzz-IEEE '97)*, Barcelona (Spain), 1997, 1339.
24. Castellano, G. et al., Reactive navigation by fuzzy control, in *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems (Fuzz-IEEE '96)*, 3, New Orleans (USA), 1996, 2143.
25. Arrúe, B.C. et al., Application of virtual perception memory to control a non-holonomic mobile robot, in *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications-SICICA '97*, Annecy (France), 1997, 537.
26. Uribe, J.P., Urzelai, J., and Ezkerra, M., Fuzzy controller for wall-following with a non-holonomous mobile robot, in *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (Fuzz-IEEE'97)*, Barcelona (Spain), 1997, 1361.
27. Ahrns, I. et al., Neural fuzzy techniques in sonar-based collision avoidance, in *Soft Computing for Intelligent Robotic Systems*, vol. 21 of *Studies in Fuzziness and Soft Computing*, Physica-Verlag, Heidelberg, 1998, 185.
28. Iglesias, R. et al., Implementation of a basic reactive behavior in mobile robotics through artificial neural networks, in *Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN '97)*, no. 1240 in Lecture Notes in Computer Science, Springer-Verlag, Lanzarote (Spain), 1997, 1364.
29. Iglesias, R. et al., Supervised reinforcement learning: Application to a wall following behavior in a mobile robot, in *Tasks and Methods in Applied Artificial Intelligence*, vol. 2 of Lecture Notes in Artificial Intelligence, 1998, 300.
30. Kohonen, T., *Self-Organizing Maps*, Springer Verlag, Heidelberg, 1997.
31. Elnagar, A. and Gupta, K., Motion prediction of moving objects based on autoregressive model, *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 28, 803, 1998.
32. Chang, C.C. and Song, K.T., Environment prediction for a mobile robot in a dynamic environment, *IEEE Transactions on Robotics and Automation*, 13, 862, 1997.
33. Spence, R. and Hutchinson, S., An integrated architecture for robot motion planning and control in the presence of obstacles with unknown trajectories, *IEEE Transactions on Systems, Man and Cybernetics*, 25, 100, 1995.
34. Chakravarthy, A. and Ghose, D., Obstacle avoidance in a dynamic environment: A collision cone approach, *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 28, 562, 1998.
35. Gil de Lamadrid, J.F. and Gini, M.L., Path tracking through uncharted moving obstacles, *IEEE Transactions on Systems, Man and Cybernetics*, 20, 1408, 1990.
36. Tsoularis, A. and Kambahmpati, C., On-line planning for collision avoidance on the nominal path, *Journal of Intelligent Robotic Systems*, 21, 327, 1998.
37. Tsoularis, A. and Kambahmpati, C., Avoiding moving obstacles by deviation from a mobile robot's nominal path, *International Journal of Robotics Research*, 18, 454, 1999.
38. Garnier, P. and Fraichard, T., A fuzzy motion controller for a car-like vehicle, Tech. Rep. 3200, INRIA, 1997.

39. Pratihar, D.K., Deb, K., and Ghosh, A., A genetic-fuzzy approach for mobile robot navigation among moving obstacles, *International Journal of Approximate Reasoning*, 20, 145, 1999.
40. Mucientes, M. et al., Use of fuzzy temporal rules for avoidance of moving obstacles in mobile robotics, in *Proceedings of the 1999 Eusflat-Estylf Joint Conference*, Mallorca (Spain), 1999, 167.
41. Mucientes, M. et al., Avoidance of mobile obstacles in real environments, in *Proceedings of the IJCAI-2001 Workshop on Reasoning with Uncertainty in Robotics*, Fox, D. and Saffiotti, A., Eds., Seattle (USA), 2001, 69.
42. Mucientes, M. et al., Fuzzy temporal rules for mobile robot guidance in dynamic environments, *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews*, 31, 391, 2001.
43. Fiorini, P., Robot Motion Planning among Moving Obstacles, Ph.D. thesis, University of California, Los Angeles, 1995.
44. Rodríguez, M. et al., Probabilistic and count methods in map building for autonomous mobile robots, in *Advances in Robot Learning*, vol. 1812 of Lecture Notes in Artificial Intelligence, Springer Verlag, Lausanne (Switzerland), 2000, 120.