Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT
ELSEVIER
Fuzzy Sets and Systems 134 (2003) 83–99
FUZZY sets and systems
www.elsevier.com/locate/fss

# A fuzzy temporal rule-based velocity controller for mobile robotics

M. Mucientes[a], R. Iglesias[a], C.V. Regueiro[b], A. Bugarín[a], S. Barro[a,*]

[a]*Department of Electronics and Computer Science. University of Santiago de Compostela, 15782 Santiago de Compostela, Spain*
[b]*Department of Electronics and Systems, University of A Coruña, 15071 A Coruña, Spain*

**Abstract**

This paper describes a velocity controller implemented on a Nomad 200 mobile robot. The controller has been developed for wall-following behaviour, and its design is modularized into two blocks: angular and linear velocity control. A simple design and implementation was made for the former, with the aim of focusing the design efforts on the linear velocity control block, in order to remark the usefulness of this task. The latter has been implemented using an explicit model for knowledge representation and reasoning called fuzzy temporal rules (FTRs). This model enables to explicitly incorporate time as a variable, due to which the evolution of variables in a temporal reference can be described. Using this mechanism we obtain linear velocity values that are adapted to each different circumstance, and thus a higher average velocity as well as smoother and more robust behaviours are achieved.

*Keywords:* Mobile robotics; Wall-following behaviour; Fuzzy temporal rules; Velocity control

## 1. Introduction

One of the main objectives of mobile robotics is the construction of autonomous systems that are capable of moving in real environments without the help of a human operator. There are numerous difficulties in the attainment of this objective, due principally to the fact that real environments are, generally, uncertain, unknown and dynamic: thus, the knowledge which is available a priori on these environments may be non-existent, incomplete, uncertain or imprecise, or the environments may undergo modifications over time, which renders them more complex and unpredictable.

All these inconveniences are compounded due to the sensorial limitations of mobile robots. For example, the use of ultrasound sensors is frequent in mobile robotics, due to their low cost with

* Corresponding author. Tel.: +34-981-563100 ext. 13560; fax: +34-981-528012.
*E-mail address:* senen@dec.usc.es (S. Barro).

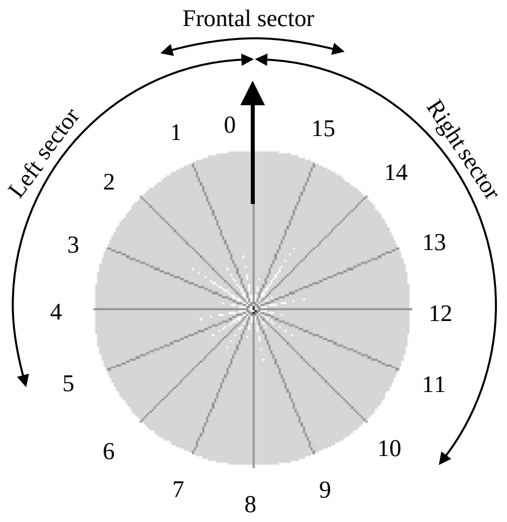

Fig. 1. Layout of the ultrasound sensors in a Nomad 200 robot. Straight arrow shows robot's moving advance direction.

respect to other solutions (laser, cameras, etc.) and due to the fact that they enable a fairly wide range of distances to be covered. Nevertheless, there is a high degree of imprecision with these sensors, due to which the perception of the environment that is obtained is less reliable, mainly in those cases in which there are objects with irregular boundaries, with smooth surfaces, gaps, etc.

For these reasons, fuzzy logic has been used as a powerful tool for dealing with this type of problems [18], given that it enables the imprecision that is inherent in the measurements obtained by the robot's sensors to be handled. One of the principal applications of fuzzy logic techniques to autonomous robots has been the design of knowledge-based controllers that perform the control task by means of *if-then* rules. In the majority of cases, these fuzzy controllers are aimed at the implementation of basic behaviour patterns [1, 10, 16] (wall following, door crossing, avoiding moving objects, etc.), which form part of a complex architecture dedicated to the achievement of an objective or higher-level task, following a plan that was previously designed. In order to do so, the different behaviour patterns are planned, activated and coordinated with the aim of reaching this objective.

One of the most frequently used behaviour patterns in robot navigation is wall following [1, 3, 15]. This behaviour is usually implemented when the robot moves between two given points in unknown environments. This paper describes a controller for wall-following behaviour which supplies the robot with suitable linear and angular velocity values at every moment. The design has been modularized aiming two objectives: firstly, to determine in which direction the robot should turn at each instant, in order that it should remain parallel to, and at a certain distance from the wall that is being followed; secondly, the maximum linear velocity at which the robot may move at each moment, without deteriorating the behaviour that is pursued, has to be determined.

The controller has been implemented on a Nomad 200 robot (Nomadic Technologies, Inc.). This has an ultrasound ring comprising 16 sensors which supply the input data of the control system. The ultrasonic sensors have a range of approximately 6.5 m, and their layout is shown in Fig. 1.

A good wall-following controller is characterized by three aspects: firstly, maintaining a suitable distance from the wall that is being followed; secondly, to move at a high velocity whenever the

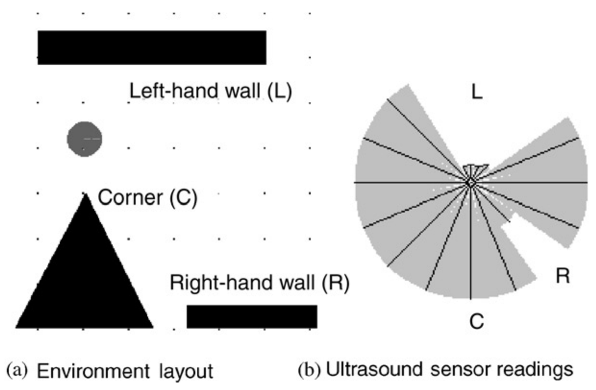

Fig. 2. Perception failure due to specular reflection when the robot approaches corner $C$.

layout of the environment is permitting, and lastly, to avoid sharp movements, making smooth and progressive turns and changes in velocity. Moreover, filtering of part of the large amount of sensorial noise has to be performed, in order to deal with possible incorrect perceptions of the environment due to unsuitable positioning of the robot. Most of the knowledge related with these features and tasks requests an explicit management of temporal evolution of input variables ("the right distance is decreasing throughout the last seconds", ...) which cannot directly be implemented by using conventional fuzzy control rules.

Different proposals have been made aiming to include temporal information in fuzzy rules [7, 9, 13, 14, 17, 19], either describing a method for performing temporal reasoning onto fuzzy propositions, or introducing time-dependent fuzzy sets, or a vague time delay between antecedents and consequents. Nevertheless, none of these approaches are capable of reasoning on the persistence of facts throughout time, which is, in our opinion, a very important source of expressiveness when managing temporal information in real-time systems (thus allowing, for example, the use of quantification, reduction or specification operators [5, 6]).

Therefore, we implemented the linear velocity controller by means of the Fuzzy Temporal Rules (FTRs) model described in [4] that enables us to collect the knowledge available in a suitable manner, as well as to determine the evolution of a variable throughout the temporal interval under study, due to which it is possible to more accurately anticipate future positions of the robot in the environment. In spite of the computational requirements being higher when using this scheme, the real time requisites of the Nomad 200 robot (three orders/s) can be met.

The other great advantage that is afforded by this FTRs model is the filtering of sensorial noise. As it is well-known, ultrasound sensors exhibit a number of problems (specular reflection, low angular resolution, "crosstalking"), which may produce distance erroneous measurements. In Fig. 2 it is shown how the robot correctly detects the straight walls $L$ and $R$, but the same is not true for $C$ corner, not detected by the sensors. This problem is compounded in real environments and occurs in a number of less extreme situations than this one.

As shown in the results section, the quality of the behaviour and the complexity of situations that this approach deals with are much higher than the ones described in other proposals [3, 8, 10], where the values of the control variables are calculated according to the values of the input variables at a single instant or reflect variations between two given instants and not throughout a particular period.

Throughout this paper special emphasis will be placed on the role played by FTRs, not only due to their special importance for resolving the proposed task in satisfactory manner, but also with the aim of showing its general interest in the development of behaviour patterns in the field of mobile robotics [16].

## 2. Description of the control system

The graphical description of the velocity control system for wall-following is shown in Fig. 3. The control system is made up of the *linear velocity control* and *angular velocity control* blocks. The calculation of the values of these control variables is carried out in a totally independent manner, in spite of the fact that they are directly linked through the turn radius. We now go on to consider each block in more detail.

### 2.1. Angular velocity control

Our angular velocity controller has a very basic and simple operational mode. The aim of this block is to determine the turn direction of the robot in order to keep it more or less parallel to the wall that is being followed, and also at an adequate distance of it. Other alternatives exist [8, 11], although we opted for the one described here due to its simplicity, as well as the fact that it leads to robust behaviour patterns. Moreover, it enables us to easily emphasize the capabilities of the linear velocity controller to guarantee that the robot should move with a suitable turn radius, as well as the usefulness and interest of FTRs in the domain of mobile robotics.

In the first place, a good state representation that reduces the large number of different situations that the ultrasound sensors may detect is described. We will see a procedure for abstracting the information coming from the frontal and lateral sensors that potentially may be perceiving both the wall which is being followed by the robot, and also frontal obstacles. In the second place, it should be considered that the greater the distance an obstacle is detected, the greater the sensorial measurements imprecision. Even more, it is possible that significant aspects of the environment may not be detected if placed beyond a threshold distance, but will be perceived when the robot gets closer to them. Because of all of these reasons we defined a six binary components vector that helps in this task in the following way: each component is associated to a particular direction with

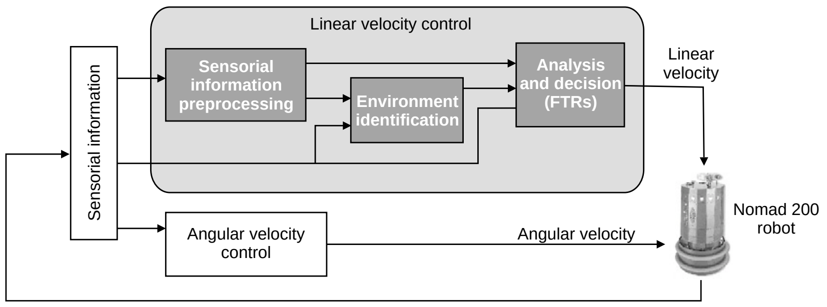

Fig. 3. Block diagram of the control system.

respect to the robot. Those components with a null value signal the presence of possible gaps, free spaces, or the existence of obstacles beyond the distance threshold previously mentioned. On the contrary, the non-null components indicate which segments of the wall or which obstacles are the closest to the robot.

We will describe how this information representation is adequate for our aim, by means of two examples: whenever the robot is placed very close to the wall, a large number of sensors detect it (under the threshold) and therefore the vector will have a large number of non-null components. Whenever the robot gets closer to a frontal wall, and provided the distance is low enough, the components of the vector associated to the frontal direction become non-null. The opposite happens whenever the robot gets away from the wall.

As a result, all the situations in which the robot may find itself, are described by means of the $2^6 = 64$ different values that the six binary components vector may take. We will use this vector as a description of the state of the environment surrounding the robot.

A multilayer perceptron network has been trained with the error backpropagation algorithm, using a small set of examples, in order to generate a table in which the angular velocity that the robot should attain in each state is stored. As a result, in our case, the network is capable of generalizing and generating the suitable action for each one of the possible 64 states.

## 3. Linear velocity control

As can be seen in Fig. 3, three modules make up the linear velocity control block: *sensorial information preprocessing*, *environment identification*, and *analysis and decision*. The *analysis and decision* module performs the task of determining the most suitable linear velocity at every instant. In order to facilitate the process of tuning this knowledge and improving the final performance of the system, three mutually independent situations are described: straight wall, open corner and closed corner. Identification of these three situations is undertaken in the *environment identification* module.

Assuming that the robot attempts to follow a wall on its right, we represent as *open corner* (see $O_i$ in Figs. 9 and 10 for a graphic example) 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. Analogously a *closed corner* situation ($C_i$ in Figs. 9 and 10) occurs for similar conditions, with the robot turning to the left. Any other situation is placed in the *straight wall* category ($S_i$ in Figs. 9 and 10). In the case of following a wall to the left, the turning will refer, in each of the aforementioned situations, to the opposite direction to the one we have just indicated.

Both the knowledge that is collected in the *analysis and decision* module, as well as the modus operandi of the processes included in the *environment identification* module, are based on two different levels of information abstraction: the measurements of distance originating from the ultrasound sensors and the information coming from the *sensorial information preprocessing* module.

### *3.1. Sensorial information preprocessing module*

Abstraction of the sensorial information provided by the ultrasound sensors of the robot that are perceiving the wall being followed or the obstacles that are close to the robot advance direction is carried out by this module. The amount of information that is taken into account is heuristically

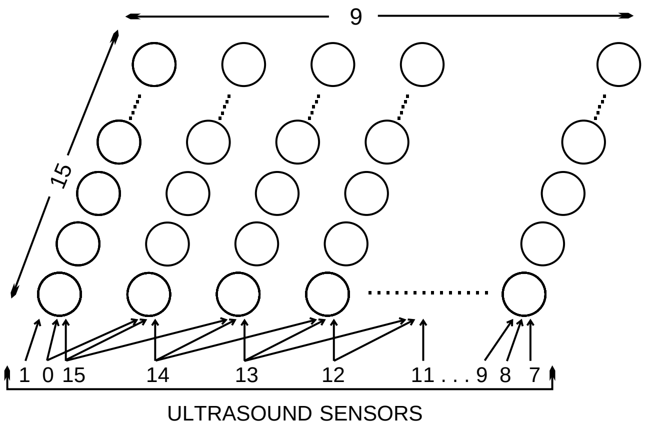

Fig. 4. Topology of the one-dimensional Kohonen networks and distribution of the sensorial inputs.

reduced, by considering 11 significant sensors. For example, whenever a right wall is being followed, sensors $1, 0, 7, 8, \ldots, 15$ (see Fig. 1) are the ones taken into account.

Self-organizing maps [12] are used in order to process this sensorial information. Instead of implementing one single network, where every neuron should process the information coming from all of the sensors, we use a number of one-dimensional networks, since this produces a better performance and reduces the complexity of the problem. As shown in Fig. 4, each of these one-dimensional nets is made up of 15 neurons and receives the measurements coming from three adjacent sensors. Since every measurement is processed by three different nets, nine of them are needed for processing all of the information coming from the 11 sensors that are taken into account.

It is important to notice that every network learns to classify what the robot "sees" in one given direction. In general, self-organizing maps can make that each neuron in the net tend to learn a position in the multidimensional space around which the patterns of a distribution are placed. The network learns to recognize which are the most frequent patterns, given that most of the positions that are learned by the neurons are located at zones where the pattern density is higher. 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 was trained, its operation occurs always in the same way: a competitive process occurs at every instant which produces one winner neuron at every network. This winner neuron will be the one for which its learned pattern is the most similar to the scenario the robot is perceiving in the direction corresponding to that network. In general, we have noticed that whenever obstacles are placed near the robot at one given direction, the winner neuron is usually one of the last neurons in the corresponding network, and vice versa. In this manner, it is possible to obtain a vector $\vec{P}$ in which each component identifies the winning neuron in each of the Kohonen networks. It can be assumed that each component of $\vec{P}$ codifies the global distance to the obstacles detected with a particular orientation with respect to the robot.

## 3.2. Environment identification module

The purpose of this module is to identify which of the three previously described situations (straight wall, open corner, closed corner) is the adequate description for the robot's current scenario.

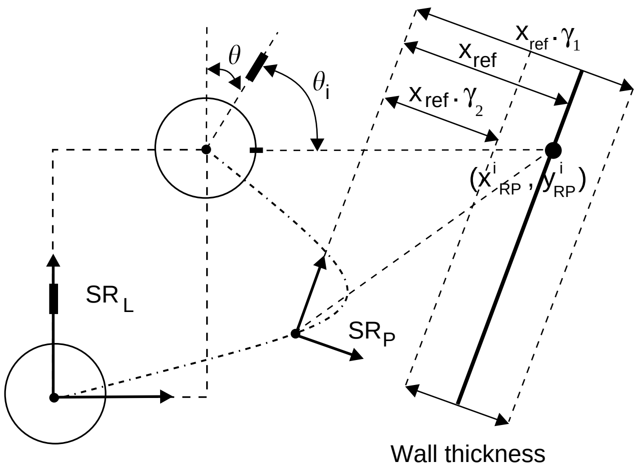

Fig. 5. Reference systems used by the *environment identification module* are shown. Through parameters $\gamma_1$ and $\gamma_2$ it is assumed that the wall of interest has a certain thickness.

Nomad 200 robot is capable of supplying, at every instant, its coordinate values with respect to a local reference system, the origin of which coincides with the starting position of the robot, and whose $y$-axis is aligned with the forward direction in the starting position ($SR_\mathrm{L}$ in Fig. 5). It is also possible to estimate, at each instant, the angle formed by each one of the ultrasound sensors with respect to the forward direction of the robot ($\theta_i$ in Fig. 5). All this information will henceforth be referred to as odometric information.

In order to identify situations of interest, whenever the robot is following a more or less straight wall (this can be deduced from the analysis of vector $\vec{P}$ and the sensorial measurements), a reference system, $SR_p$, (which is characterized by presenting the $y$-axis, orientated, as far as is possible, in a parallel manner to the wall) is constantly updated. All points belonging to the wall that is being followed will be characterized in this reference system by having the same value for the $x$ coordinate, denoted as $x_\mathrm{ref}$.

By using the odometric information, it is possible to translate each of the sensorial readings into a position in $SR_p$, represented by $(x^i_{RP}, y^i_{RP})$, where the index $i$ identifies a particular sensor. Therefore, for all those sensors that are perceiving the same wall as the one being followed, coordinate $x^i_{RP}$ will be very similar to $x_\mathrm{ref}$. In turn, when coordinates associated to the wall being followed stop being received, the vector $\vec{P}$ from the *sensorial information preprocessing module* will aid us in verifying whether this is due to the presence of an open or closed corner.

Without going into the operational details of this module, given that its description remains outside the scope of the present work, we should emphasize the presence of certain problems that have been taken into consideration. Firstly, it should be mentioned that there are certain errors that are associated to the odometric information. These errors, along with the sensorial noise itself, hinder a precise calculation of coordinates $(x^i_{RP}, y^i_{RP})$. In order to reduce the influence of this type of error, we assume that the wall of interest has a certain thickness (Fig. 5), i.e., it is possible to identify the sonars that perceive the wall as those for which it is fulfilled that $\gamma_2 \leqslant x^i_{RP}(t)/x_\mathrm{ref} \leqslant \gamma_1$. Values for parameters $\gamma_1 \geqslant 1$ and $\gamma_2 \leqslant 1$ may be established a priori by the task designer, and may even be modified according to the variation of the dispersion observed in the $x^i_{RP}$.

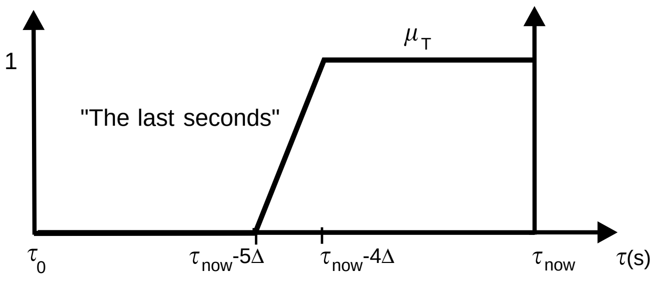

Fig. 6. Temporal reference $\mu_T$ associated with the expression "*the last seconds*".

On the other hand, in order to obtain a robust identification of the situation in which the robot finds itself, wherever possible, the information corresponding to the current sensorial readings tends to be merged with that coming from recent previous measurements.

Given that the wall that is being followed may not be perfectly straight, the reference system associated to this wall ($SR_p$) is recalibrated each time the robot has overtaken a closed/open corner situation and also whenever it reaches an almost parallel position with respect to the wall. This recalibration contributes to minimizing the effects of the odometric errors. For similar reasons, the parameter $x_{\mathrm{ref}}$ is also frequently updated.

Finally, a truth value $\mu_i(t)$ is associated to each one of the sensorial readings, which depends on the angle of incidence, i.e., the angle formed by the axis of the corresponding sonar with the $x$-axis of $SR_p$. In this way, those readings for which this angle has a high value will have a lower degree of credibility and influence on the modification of parameters.

### 3.3. Analysis and decision module

The *analysis and decision* module is essential 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.

Since knowledge representation and reasoning in this module is made using FTRs, we will firstly give a brief description of the underlying temporal model we used.

#### 3.3.1. Temporal reasoning: fuzzy temporal rules model

In the FTRs model we have used for this implementation [2,4,5] propositions are of the form $X$ *is* $A$ $\langle$*in* $Q$ *of*$\rangle$ $T$, 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, being in both cases membership functions defined on a discrete set of values $\tau = \{\tau_0, \tau_1, \ldots, \tau_k, \ldots\}$, where each $\tau_k$ represents a precise temporal instant and $\tau_0$ represents the origin. We assume that the values of this set are evenly spaced, where $\triangle = \tau_j - \tau_{j-1}$ is the unit of time, whose size or granularity depends on the temporal dynamics of the application that is being dealt with. For this application $\triangle = \frac{1}{3}s$, which is the time that passes between two consecutive control orders. Temporal distributions are defined in relation to the current temporal point, $\tau_{\mathrm{now}}$, as shown in Fig. 6.

Different scenarios can occur for the fulfilment of non-temporal part "*X is A*" when $T$ is an interval: non-persistence (as in "*velocity is high in the last seconds*"), where fulfilment is required for at least one point of $T$, persistence (as in "*velocity is high throughout the last seconds*"), where fulfilment is required throughout the entire interval, and partial persistence, where the non-temporal part should be fulfilled for some subinterval ("*in the majority of T*", "*in part of T*").

The execution process of a FTR differs from that of a conventional fuzzy rule in the calculation of the degree of fulfilment (DOF), which now also depends on the prior values of variables. The calculation of DOF is carried in the model in the following manner: in the first place the degree of fulfilment of the non-temporal part of the proposition is calculated, which is defined as

$$sc(\tau_k) = \mu_A(X(\tau_k)), \quad \tau_k \in \tau, \tag{1}$$

where $\mu_A$ is the membership function that is associated to the value $A$ of the proposition, and $X(\tau_k)$ is the value observed for the variable $X$ at the temporal point $\tau_k$.

Secondly, $sc$ is modulated by the temporal part of the proposition, so that in all the three cases the weight that is given to the temporal points is proportional to its membership in $T$ ($\mu_T$):

- *Non-persistence*: "*X is A in T*"

$$DOF = \bigvee_{\tau_k \in \tau} sc(\tau_k) \wedge \mu_T(\tau_k), \tag{2}$$

- *Persistence*: "*X is A throughout T*"

$$DOF = \bigwedge_{\tau_k \in \tau} sc(\tau_k) \vee (1 - \mu_T(\tau_k)), \tag{3}$$

- *Partial persistence*: "*X is A in Q of T*"

$$DOF = \mu_Q \left( \frac{\sum_{\tau_k \in \tau} sc(\tau_k) \wedge \mu_T(\tau_k)}{\sum_{\tau_k \in \tau} \mu_T(\tau_k)} \right). \tag{4}$$

The operators $\wedge$ and $\vee$ are, respectively, the t-norm minimum and the t-conorm maximum, and $\mu_Q$ is the membership function that is associated to the linguistic quantifier $Q$.

Fig. 7 shows some definitions for the membership functions $\mu_Q$ associated to the temporal persistence quantifiers used for the linear velocity control of the robot ("*in*" "*approximately in*", "*throughout*" and "*in part of*"). Moreover, the temporal references used in the rules are made up of the last 2, 3 or 4 measurements. The use of larger intervals has been avoided, since the dynamics of the application to be controlled is extremely fast.

This model of rules representation and reasoning has been used in order to characterize each one of the situations of interest in the problem of linear velocity control (straight wall, open corner and closed corner). Each one of these has entailed the design of an independent fuzzy temporal knowledge base (KB), which we now go on to describe in the following sections.

### *3.3.2. Straight wall situation*

The rules of this KB have been designed with the aim of making 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

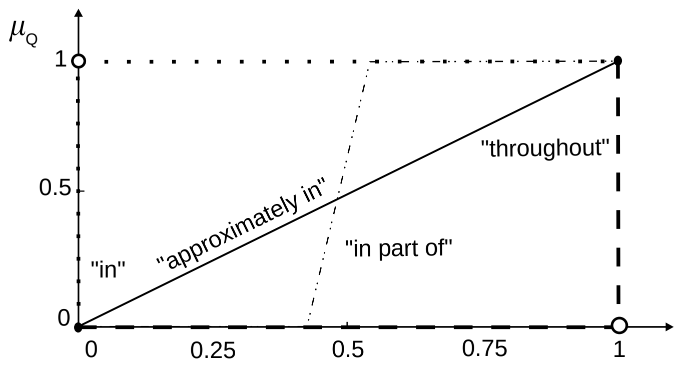

Fig. 7. Membership functions ($\mu_Q$) of the temporal quantifiers used.

(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 appropriately reduced. For example, the existence of a wall or obstacles situated on the left-hand side of the robot, and relatively close to it, may result in the robot having to move closer to the right-hand wall in order to move away from the obstacles to the left. Thus, for low or medium distances to the left the robot's speed will be significantly reduced, as it is necessary to move closer to the right-hand wall.

Assuming that the robot follows the right-hand wall, the fundamental variable when selecting the most suitable speed will be the distance to that wall, defined as the minimum distance of the sensors in the right sector (Fig. 1). The robot position must also influence the calculation of velocity. This information is picked up by the variable *distances quotient*:

$$distances\ quotient = \frac{distance_{\mathrm{left}}}{distance_{\mathrm{right}}}, \tag{5}$$

where the left-hand distance is calculated as the minimum distance provided by the left sector sensors. Although the distance to the right-hand wall often undergoes variations that do not lead to a change in the linguistic label, it seems clear that a good controller should consider minor changes in the values for this variable, in order to obtain a smoother and more continuous behaviour. For this reason, variable *trend of the right-hand distance* is calculated on the basis of the current and previous values of the right-hand distance:

$$trend_{\mathrm{right\,dist}}(\tau) = dist_{\mathrm{right}}(\tau) - dist_{\mathrm{right}}(\tau - 1). \tag{6}$$

On the other hand, situations occur for which orientation of the robot with respect to the right-hand wall is needed to be known. This may happen because of the proximity of the wall or to the magnitude of the change in velocity which it is desired to implement. For this reason, variable *orientation* is defined for evaluating whether the robot is facing the wall being followed or not:

$$orientation = \frac{\max\{P_1, P_2, P_3\}}{\max\{P_4, P_5, P_6\}}, \tag{7}$$

where $P_i$ is the $i$th component of vector $\vec{P}$ described in the *sensorial information preprocessing module*. This variable is used as a complementary aid mainly when $trend_{\mathrm{right\,dist}}$ is stable, in order to have a better description of the current situation of the robot. Low values for *orientation* indicates that the robot is not facing the wall of interest. We now go on to show an example of a rule from this KB:

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

Since variables obtained from the readings of the ultrasonic sensors (*distances quotient*, *right distance* and $trend_{\text{right dist}}$) may indeed contain noise due to the reflections of the ultrasonic beams, carrying out a temporal processing of this information ("the last three measurements", etc.) helps to avoid spurious erroneous data, which could cause unnecessary alterations in the velocity, enhancing the robustness of the behaviour pattern. Furthermore, evolution of the distance to the right-hand wall can be analysed, in order to determine its tendency, which 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 behaviour pattern, as the controller will anticipate future positions of the robot with respect to obstacles, thus avoiding sharp changes of velocity.

Moreover, the use of FTRs provides a great deal of flexibility to the 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 in order that the fuzzy temporal proposition be fulfilled. Thus, in the example rule, quantifier "*approximately in*" means that the degree of fulfilment of the proposition is proportional to the percentage of points in the temporal interval that fulfil the spatial part.

### 3.3.3. Open corner situation

This situation is defined as that in which the robot has to turn to the right, with a low turning radius, and tracing an arc of almost 90° or over ($O_i$ in Fig. 9). This situation is truly complex due to two factors: on one hand, low velocities may be needed in order to produce low turning radius values. On the other hand, the robot's sensors may cease to detect the right-hand wall correctly at some given moments, even though some of them may detect it occasionally (Fig. 2). FTRs are suitable tools for tackling this problem, as even though perception may have been unsatisfactory in some temporal points, the analysis of the variables in a temporal interval can make the system's response globally correct.

It is assumed that the robot needs to trace a circumference arc that is centred 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:

$$v_{\text{desired}} = v_{\text{angular}} \times radius_{\text{desired}}, \quad \text{where } radius_{\text{desired}} = 50\,\text{cm}.$$

Given that $v_{\text{desired}}$ is just a value that is close to the desired one and which is taken as a reference, it is advisable, as a precaution, that the linear velocity at which the robot moves initially should be slightly under that one, so that the robot should not move too far away from the wall. Nevertheless, as the robot advances, a turning radius that is lower than the desired one may lead the robot to end up facing or too close to the wall. In order to avoid this, velocity should be increased slightly, which will be reinforced by the fact that the probability that the open corner should finish at any moment increases. For all these reasons, the angle turned by the robot from the beginning of the corner is considered in the antecedent part of the rules:

$$angle_{\text{turned}} = angle_{\text{initial}} - angle_{\text{current}}. \tag{8}$$

Finally, using the right-hand distance, it is possible to detect the presence of a nearby wall which could be considered as the end of the open corner. Since it is likely that the robot may have moved with fairly unreliable information in the final instants, it is necessary to verify that its orientation is suitable. For this reason, the *orientation* variable and its tendency also influence in the selection of the behaviour pattern. For example, even when robot's orientation has been suitable (robot not facing the wall), if its tendency indicates that the robot is facing the wall it will not be advisable to increase the linear velocity, although the distance to the wall is correct, given that an increase would take it closer to the wall. A typical rule from this KB, which comes into action in the final instants of an open corner, is:

IF ***right-hand distance*** *is low approximately in the last four measurements* AND ***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 ***orientation*** *is improving approximately in the last four measurements* AND ***velocity*** *is medium* THEN *increase* ***velocity*** *a little.*

### 3.3.4. Closed corner situation

The final situation to be dealt with is that of a closed corner. Let us remember that this situation is characterized by the robot turning to the left with a low turning radius ($C_i$ in Fig. 9). As opposed to the other situations, a closed corner situation is associated to the presence of a frontal object that is placed at the end of the straight wall that is being followed. For this reason, the frontal distance (calculated as the minimum distance of the frontal sector sensors) becomes one of the relevant variables that determines the robot's behaviour, even when it is also necessary to take into account the distances to the right and to the left. In order to achieve more precise control, when one of the distances has a low value its trend is also analysed, in order to be able to thus detect tendencies in the evolution of the values of these variables.

Whilst the robot is still a long way from the corner, its behaviour 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 it is possible for the robot to have to turn even up to 180° for some cases. This depends on whether obstacles are situated to the left of the robot and close to the corner. These very different situations can be identified by the fact that, once the robot reduces speed and starts turning, the wall placed on the left will now become a frontal wall. In order 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. Only when this quotient increases significantly it is possible to assume that the corner has come to an end. An example of this situation is found in the following rule:

IF ***frontal distance*** *is low in part of the last four measurements* AND ***frontal distance*** *is increasing approximately in the last four measurements* AND ***quotient between the left-hand and frontal***

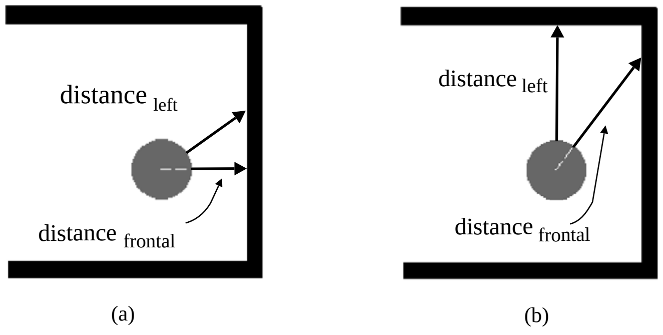

Fig. 8. Utilization of the quotient between the left and frontal distances: $quotient_a \simeq quotient_b$, $distance_{frontal\_a} \ll distance_{frontal\_b}$.

***distances*** *is low in part of the last four measurements* AND ***velocity*** *is low* THEN *reduce* ***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, since "*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 (Fig. 8).

## 4. Results

In order to verify the operation of the system a number of experimental tests have been carried out, both in real and simulated environments, with highly satisfactory results. Despite the large number of rules needed for improving the behaviour in any kind of situation (108, 140 and 65 for straight wall, open corner and closed corner, respectively), the execution time per cycle of the global control system is 0.22 s, which allows ample margin for carrying out other tasks, since control actions have to be taken every third of second.

In this section we describe two examples[1] carried out on the real robot,[2] including the values of certain objective parameters which endorse the global operation of the system (average velocity and average distances).

In the first example (Fig. 9) a complex environment with a high number of closed corners ($C_i$), open corners ($O_i$), gaps, doors, etc. is shown. In order to show the complexity of the proposed task, in the same figure we also show the readings collected by the ultrasound sensors whilst the robot was in motion. On the other hand, by means of the robot's trajectory (represented by circles),

[1] Video recordings of these and other tests are available at http://www-gsi.dec.usc.es/areas/robotica/fss01.htm.

[2] The most important characteristics of the Nomad 200 robot are a diameter of 53 cm, a maximum linear velocity of 60 cm/s (in this case, limited to 45 cm/s), a maximum angular velocity of 45°/s, and an ultrasound range of between 15 and 647 cm.

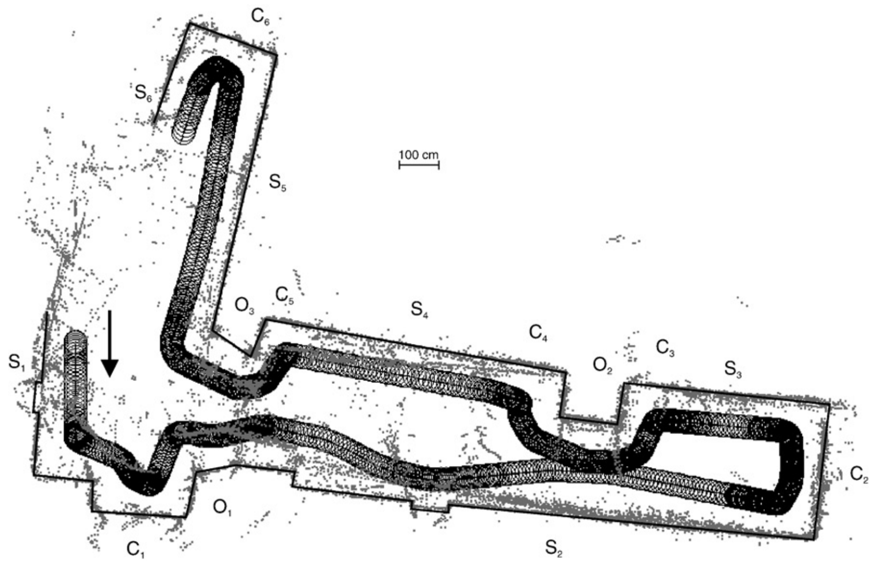

Fig. 9. Example 1. See text for description.

Table 1
Data of the two examples described

| | Example 1 | Example 2 |
|---|---|---|
| Average right distance (cm) | 47 | 49 |
| Average left distance (cm) | 131 | 127 |
| Average velocity (cm/s) | 17 | 25 |
| Maximum velocity (cm/s) | 41 | 44 |
| Minimum velocity (cm/s) | 1 | 0 |
| Time spent | 3′51″ | 3′55″ |

information is implicitly included on its velocity at each instant (a higher the concentration of marks indicates lower velocity).

The first point that should be emphasized is the high reliability and robustness of the control system, since the average distance from the right-hand wall was 47 cm. On the other hand, it can also be seen how the system is capable of satisfactorily resolving all situations, increasing velocity when following a straight wall ($S_2$) 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 ($C_2, C_6$) in which a turn of about 180° must be realized, or when it is not necessary to reduce speed so drastically due to corner not being so pronounced ($C_3, C_4$) and a 90° turn is sufficient. Although the average velocity in this example (17 cm/s) may be apparently considered as low, this result is justified by the high complexity of the environment, which has a large number of corners. The *angular velocity control* block proposes extremely low angular velocities at these corners, which forces the linear velocity control system to reduce speed in order to maintain a low turn radius.

Table 1 shows some of the data measured in the two tests described. It can be seen how the average velocity obtained in the example shown in Fig. 9 (example 1) is lower than the one obtained

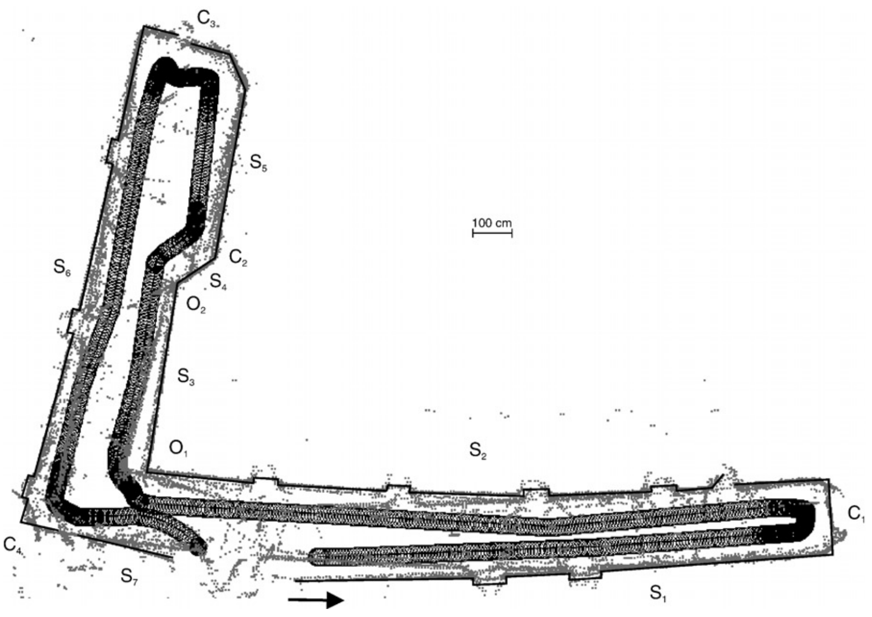

Fig. 10. Example 2. See text for description.

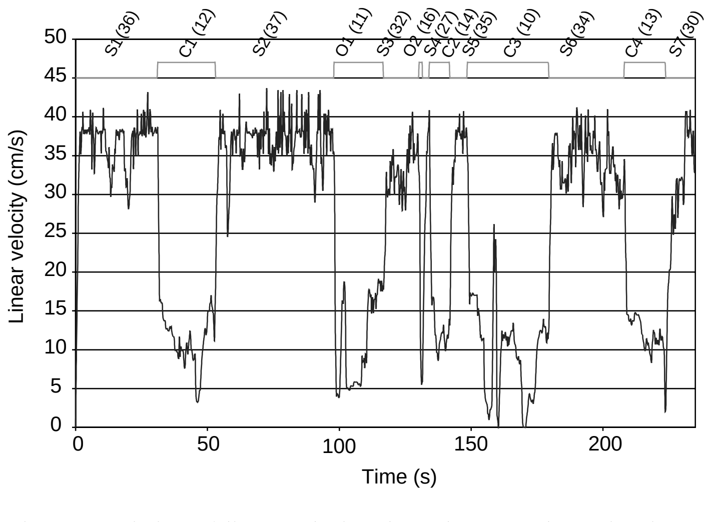

Fig. 11. Variation of linear velocity along the route shown in Fig. 10.

during the test shown in Fig. 10 (example 2) due to the complexity of the environment in the first example. The maximum and minimum velocities obtained in the two environments are identical, it being noteworthy that in all two cases, the robot, at some point, has to come to a halt in order to facilitate the turn, thus average velocities are not higher. Average right distance always lies within the requested 45–50 cm interval, whilst average left distance is higher.

One highly noteworthy aspect is the controller's capability to adapt the linear velocity (by increasing or decreasing it), according to needs. This aspect can be evaluated in Fig. 11, which shows the evolution of velocity as opposed to time in the route shown in Fig. 10.

In the diagram, each one of the situations of the environment are labelled in the upper section, as well as the average velocity in each of the sections (in cm/s). It can be seen how for straight walls the robot reaches and maintains the maximum speed permitted (some 40 and 30 cm/s average) for sufficiently long walls ($S_1$, $S_2$, $S_3$, $S_5$ and $S_6$). At $S_2$ velocity is reduced in the first instants due to the presence of a half-open door at the beginning of the straight wall. At closed corners the range of velocities is logically much lower than in the straight wall situation (average speeds close to 12 cm/s). In situations such as $C_2$, where the robot must turn less than 90°, the minimum velocity is 9 cm/s, while in very closed corners (180°) such as $C_3$ in some cycles the robot must stop in order to realize the turn, consequently reducing the final average velocity. Lastly, the speed patterns for the two open corner situations are very similar: initially, velocity is low, and as the robot turns and detects the wall immediately after the corner, speed increases.

## 5. Conclusions

This work describes the design and real implementation of velocity control in wall-following behaviour in a Nomad 200 mobile robot. Its modularization into two independent control blocks, where the first of these acts on the angular velocities and the second one on linear velocities, has given rise to robust, reliable and progressive behaviours, as is evident with the numerous and complex tests that have been carried out, both in real and simulated environments.

It is important to stress the role played by the linear velocity control block. Here the knowledge has been modularized for taking into account three differentiated situations of interest: straight wall, open corner, closed corner. FTRs are capable of filtering sensorial noise, of endowing the rules with a high degree of expressiveness and of analysing the evolution of variables whilst taking past values into account, and this has noticeably improved the robustness and reliability of the system.

This block enables the linear velocity to be suitably adapted to the circumstances of the environment, even when at a certain instant the angular velocity may not be wholly suitable. Usefulness of FTRs in the field of behaviours implementation in mobile robotics, characterized by a lack of precision in measurements and the need to evaluate variables and/or tendencies within temporal intervals, is also highlighted.

## Acknowledgements

Authors wish to acknowledge support from the Secretaría Xeral de I+D (Xunta de Galicia), the Spanish Ministry of Education and Culture (CICyT) and the European Commission, and CICyT through grants PGIDT99PXI20603A, 1FD97-0183 and TIC2000-0873, respectively.

## References

[1] B.C. Arrúe, F. Cuesta, R. Braunstingl, A. Ollero, Fuzzy behaviours combination to control a non-holonomic robot using virtual perception memory, Proc. Fuzz-IEEE'97, 1997, pp. 1239–1244.

[2] S. Barro, A. Bugarín, P. Cariñena, P. Félix, S. Fraga, Petri nets for fuzzy reasoning on dynamic systems, Proc. 7th IFSA World Congress, 1997, pp. 279–284.

[3] R. Braunstingl, J. Mujika, J.P. Uribe, A wall following robot with a fuzzy logic controller optimized by a genetic algorithm, Proc. Fuzz-IEEE'95, Vol. 5, 1995, pp. 77–82.

[4] A. Bugarín, P. Cariñena, P. Félix, S. Barro, Reasoning with fuzzy temporal rules on Petri nets, in: J. Cardoso, H. Camargo (Eds.), Fuzziness in Petri Nets, Studies in Fuzziness and Soft Computing, Vol. 22, Physica-Verlag, Wurzburg, 1999, pp. 174–202.

[5] P. Cariñena, A. Bugarín, M. Mucientes, F. Díaz-Hermida, S. Barro, A model of fuzzy temporal rules for knowledge representation and reasoning, Proc. IPMU'2000, Vol. 2, 2000, pp. 1239–1246.

[6] P. Cariñena, A. Bugarín, M. Mucientes, F. Díaz-Hermida, S. Barro, Fuzzy temporal rules: a rule-based approach for fuzzy temporal knowledge representation and reasoning, in: B. Bouchon-Meunier, J. Gutierrez-Rios, L. Magdalena, R.R. Yager (Eds.), Technologies for Constructing Intelligent Systems 2: Tools, Studies in Fuzziness and Soft Computing, Vol. 90, Springer-Verlag, Heidelberg, 2002, pp. 237–250.

[7] B. Carse, T.C. Fogarty, A. Munro, Artificial evolution of fuzzy rule bases which represent time: a temporal fuzzy classifier system, Internat. J. Intell. Systems 13 (1998) 905–927.

[8] G. Castellano, G. Attolico, A. Distante, Automatic generation of fuzzy rules for reactive robot controllers, Robotics Autonomous Systems 22 (1997) 133–149.

[9] M.A. Gadeo, J.R. Velasco, Faded temporal fuzzy logic controller and temporal fuzzy logic controller. Comparative study, Proc. IPMU'2000, Vol. 2, 2000, pp. 732–737.

[10] A. García-Cerezo, A. Mandow, M.J. López-Baldán, Fuzzy modelling operator navigation behaviours, Proc. Fuzz-IEEE'97, 1997, pp. 1339–1345.

[11] R. Iglesias, C.V. Regueiro, J. Correa, S. Barro, Supervised reinforcement learning: application to a wall following behaviour in a mobile robot, in: A. Pasqual del Pobil, J. Mira, M. Ali (Eds.), Tasks and Methods in Applied Artificial Intelligence (IEA-98-AIE), Lecture Notes in Computer Science, Vol. 2, Springer, Berlin, 1998, pp. 300–309.

[12] T. Kohonen, Self-Organizing Maps, Springer, Berlin, 1997.

[13] H. Maeda, S. Asaoka, S. Murakami, Dynamical fuzzy reasoning and its application to system modeling, Fuzzy Sets and Systems 80 (1996) 101–109.

[14] O. Màs, P. Palà, J.M. Miró, Razonamiento aproximado en presencia de conceptos temporales difusos, Proc. ESTYLF'96, 1996, pp. 185–196.

[15] M. Mucientes, R. Iglesias, C.V. Regueiro, A. Bugarín, S. Barro, Control de la velocidad de un robot móvil mediante reglas temporales borrosas, Proc. ESTYLF'2000, 2000, pp. 127–132.

[16] M. Mucientes, R. Iglesias, C.V. Regueiro, A. Bugarín, P. Cariñena, S. Barro, Use of fuzzy temporal rules for avoidance of moving obstacles in mobile robotics, Proc. EUSFLAT'99, 1999, pp. 167–170.

[17] D.Q. Qian, Representation and use of imprecise temporal knowledge in dynamic systems, Fuzzy Sets and Systems 50 (1992) 59–77.

[18] A. Saffiotti, The uses of fuzzy logic in autonomous robot navigation, Soft Comput. 1 (4) (1997) 180–197.

[19] J. Virant, N. Zimic, Attention to time in fuzzy logic, Fuzzy Sets and Systems 82 (1996) 39–49.