ELSEVIER International Journal of Approximate Reasoning 25 (2000) 255–289
INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

Fuzzy behaviors for mobile robot navigation: design, coordination and fusion ☆

Eugenio Aguirre, Antonio González *

Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. de Ingeniería Informática, Universidad de Granada, 18071 Granada, Spain

Received 1 January 2000; received in revised form 1 June 2000; accepted 1 July 2000

Abstract

The implementation of complex behavior generation for artificial systems can be overcome by decomposing the global tasks into simpler, well-specified behaviors which are easier to design and can be tuned independently of each other. Robot behavior can be implemented as a set of fuzzy rules which mimic expert knowledge in specific tasks in order to model expert knowledge. These behaviors are included in the lowest level of a hybrid deliberative–reactive architecture which is aimed at an efficient integration of planning and reactive control. In this work, we briefly present the architecture and attention is focused on the design, coordination and fusion of the elementary behaviors. The design is based on regulatory control using fuzzy logic control and the coordination is defined by fuzzy metarules which define the context of applicability for each behavior. Regarding action fusion, two combination methods for fusing the preferences from each behavior are used in the experiments. In order to validate the system, several measures are also proposed, and thus the performance of the architecture and combination/arbitration algorithms have been demonstrated in both the simulated and the real world. The robot achieves every control objective and the trajectory is smooth in spite of the interaction between several behaviors, unexpected obstacles and the presence of noisy data. When the results of the experimentation from both methods are taken into account, the influence of the combination method appears to be of prime importance when attempting to achieve the best trade-off among the preferences of every behavior.

☆ This work has been supported by the CICYT under Project TAP99-0535-C02-01.

* Corresponding author. Tel.: +34-958-246143; fax: +34-958-243317.
E-mail address: a.gonzalez@decsai.ugr.es (A. González).

0888-613X/00/$ - see front matter
PII: S0888-613X(00)00056-6

Keywords: Fuzzy behaviors; Combination methods; Behavior-based systems; Autonomous robots

1. Introduction

Paradigms of intelligent behavior generation have evolved greatly since the first approaches. Focus was initially placed on known and structured environments [18] but afterwards the emphasis shifted towards approaches which were able to account for incomplete and uncertain knowledge and also to deal with the inaccuracy and imprecision inherent in robot navigation tasks in real-life situations [9]. In the first models, the process of planning alternatives in order to solve problems was based on global and explicit representations of the robot-environment and on the relationship between them. However, there was a need to incorporate reactive skills which would allow the robots to have reflexes in order to deal with real sensors and environmental inconsistencies and to recover instantaneously from unexpected events. The mobile robots have therefore had to incorporate reactive capabilities. A mobile robot that carries out navigation tasks in an office-like environment, navigating from one room to another, also needs to reason out its relationship with the environment in order to compute the best path between the current and the goal position. Therefore, deliberative reasoning is needed to elaborate abstract plans and to include decision-making mechanisms which permit the proposed tasks to be optimized. Several approaches for integrating planned and reactive behaviors have been proposed in recent years [4,16,21,27].

In this paper, we deal with a hybrid deliberative–reactive architecture for mobile robot navigation for integrating planning and reactive control and attention is focused on the design, coordination and fusion of the elementary behaviors. The complex behavior is therefore generated by combining simpler behaviors. Some approaches have been already proposed for behavior combination in mobile robots, modeled either by crisp [2,14] or fuzzy algorithms, [11,12,20,23]. Complex behavior in the robot results from the interaction between the goals, the internal states and the environment. Simple rules, which are either fuzzy or crisp, link basic or elementary behaviors to generate more complex observable behaviors.

In this work, elementary behaviors are built up as fuzzy knowledge bases since fuzzy logic offers many advantages when dealing with the uncertainty and vagueness of the robot sensors [26]. We present a methodology for designing fuzzy behaviors based on regulatory control and propose several measures in order to evaluate the robot motion. Additionally, two different fuzzy methods of combining these simpler behaviors are explained. The idea of the first combination method is to combine the preferences from each behavior before

defuzzification by means of the intersection of the outputs. Regarding the second method, the final output is obtained by a linear combination of each defuzzified fuzzy output. The study of the robot motion is based on two main measures: the achievement of the control objective and the total bending energy (TBE) of the trajectory. The objective of the first measure is to test the quality of control when the robot has reached the control objective and also during the whole run. The total bending energy, on the other hand, evaluates both the smoothness and the length of the trajectory.

The architecture is demonstrated by performing navigation tasks, first in simulated environments and later in the real world in an office-like environment with a Nomad 200 mobile robot. The results of the experiments show that the robot achieves every objective of control from every behavior and the trajectory is a smooth trajectory in spite of the interaction between several behaviors. As regards the combination methods, our experimentation shows that the influence of the combination method may be an important factor because in some cases there can be significant differences in the trajectory of the robot according to the combination method used. These measures show that the method which combines the outputs before defuzzification has better results than the method which combines the defuzzified outputs.

This paper has been organized in the following way. First, we introduce the architecture for mobile robot navigation and the methodology for designing fuzzy behaviors. This section also contains a brief description of the fuzzy behaviors used. Then, the two combination methods are explained, showing the main difference between them. Section 5 describes the fuzzy metarules which choose the context that will be activated. Afterwards, the evaluation measures are proposed and the results of the experimentation with a Nomad 200 mobile robot are described, with emphasis on reaching the control objective and the bending energy (BE) of the trajectory. Finally, the work is concluded with a discussion concerning combination methods and the performance of the system.

2. A hybrid deliberative–reactive architecture for mobile robot navigation

Our proposal is along the lines of behavior-based robots [5] and uses a three-level architecture [17] to integrate deliberative techniques and reactive behaviors. This architecture is composed of three hierarchical layers: a planning, an executive and a control level. The highest level must search for a safe and minimum-cost path from an initial position to a final desired position across an office-like environment which is expressed by means of a map that contains topological and geometrical information about the environment. This level produces a high level abstraction plan since the executive level is responsible

for achieving the abstract goals through the combination of several basic behaviors. This plan can be seen as a linguistic description of the path from initial to final position such as "follow left wall, cross the door, turn left and follow the corridor, ..." Regarding the other levels, the executive level must ensure the fulfillment of the plan by selecting which basic behaviors should be activated at a given time depending on both the environment and the current goal. Thus, this level defines the context of applicability of certain behaviors using a set of metarules. Furthermore, this level constantly monitors the performance of the robot so that any failure can be detected, and allows the system to recover from any failure. The lowest level deals with the control of the robot motion, coupling sensors to actuators. The control level is composed of several rule-based basic behaviors which can be combined to generate a more complex observable behavior. Regarding this architecture, we only focus on the control level in this paper since the behaviors are included in this level.

Typical behaviors such as the following of the corridor or wall can be considered as a control problem, which can be solved by either a classic PID or a fuzzy controller. The fuzzy controller takes advantage of expert control knowledge expressed as a set of linguistic expressions which are closer to those used by human beings. Such a control offers a satisfactory performance of the system and a high robustness in spite of noise and disturbances [10].

However, the design of the fuzzy knowledge base for a fuzzy controller demands a long optimization process since this fuzzy controller must deal with very different control situations: following a wall, following a corridor, avoiding an obstacle, etc. Rules must be designed so that they can handle many different robot-environment states and we should bear in mind that the linguistic variables can be quite different from each other according to the robot-environment state and the current goal. A way of getting round this problem is to create an initial or basic controller which can be incrementally updated and optimized by adequately tuning both labels and rules.

Another way of solving this problem involves decomposing the global behavior into several control schema or motion control descriptions [3]. The control schema can be merged by combining the corresponding actions as derived from their control rules [27]. However, such control actions may be in conflict, thus requiring an association between each control schema and its own context of applicability which is defined by means of a set of metarules that are generated by the executive level.

Within each control schema, it is possible to define the most appropriate semantics for the linguistic variables which are manipulated in the rules, and thus each behavior can be tuned independently to be more effective in its own context, and complex behaviors can be obtained based upon the composition of former simpler behaviors. This approach is more suitable from the viewpoint of the definition of fitness functions for an evolutionary learning process since the fitness function can be adapted to each elementary behavior instead of

a large complex one [7,19]. The next problem we face is the combination of simpler behaviors [8]. The most popular approaches are based on combining the outputs by means of a linear combination of the numerical outputs of each simpler behavior weighted by a value that represents the weight of the behavior, depending on the current context [6,22]. In [27] however, the outputs of both behaviors are combined before any control action is selected, and when an action is dominated by the other one, the final output will be the last one. In [25] Saffiotti splits the problem of behavior coordination into two different subproblems: how to decide which behaviors should be activated at each moment and how to combine the results from different behaviors into one command. These are known as the behavior arbitration problem and the command fusion problem, respectively. In this paper, the first problem is resolved by means of the generation of metarules and the second one by means of the combination methods.

3. A methodology for designing fuzzy behaviors

The use of fuzzy logic in the design of navigation behaviors is nowadays quite popular. The set of behaviors that are being implemented can include, for example, the following of walls, corridors or the avoidance of obstacles. There is not however an established way of designing the rule bases of these behaviors. A lot of approaches use expert knowledge to decide on the response of the behavior according to its objective but without defining that objective explicitly. On the other hand, we think that the robot must use several abstraction levels on the information that it has collected from the environment which it can use to control its own motion. Our methodology regarding the design of the rule bases for the behaviors, therefore has the following features:

- The situation to achieve or to maintain one behavior will be defined explicitly by means of certain values of the input variables.
- The regulatory control is used to build the rule base of the behavior, defining a set-point or objective of the behavior.
- The structure of the behaviors becomes very homogeneous and makes the definition of the evaluation measures shown in Section 6.1 possible.

Regarding the use of several abstraction levels on the information, we are to classify the behaviors according to the kind of input information that is used. Thus, the input data can be:

- Input data from the robot sensors with a simple pre-processing to avoid noisy data. There is no world modeling. Within this kind of behaviors we distinguish between:
  1. Behaviors addressed to reach and maintain an objective such as *Follow wall*. We call these as *Objective-oriented* behaviors.

2. Behaviors that tightly couple perception to action such as *Avoid obstacle*. These are *Purely reactive* behaviors.

- Input data from a sensor-derived world modeling. There is a world representation but only the information necessary for the performance of a specific behavior is represented. For example *Cross door* is a behavior of this kind and we call these as *Short-memory* behaviors.

The behaviors deal with the control of the robot motion, coupling inputs to actuators according to the preference of the behavior. Every behavior has been implemented as a set of fuzzy rules by means of the proposed methodology and the structure of this rule base will be independent of the kind of input information that is needed to execute the behavior.

Before explaining the design of each type of behavior, we will define the conceptual framework adopted to represent the fuzzy behaviors. The output of each fuzzy behavior in a robot-environment state is a fuzzy set $B_j(x,y)$ (Eq. (1)) derived from the fuzzy inference. Let x be the input array that defines the actual state and y the possible values in the output variable, for example the steering velocity, which shows the possible actions, then the fuzzy set B_j is given by

$$B_j(x,y) = \max_{1 \leqslant i \leqslant n} \min(Q_i(x), A_i(y)), \tag{1}$$

where $Q_i(x)$ is the truth value of the fuzzy proposition Q_i in the state x. A_i is a linguistic label of the consequent variable which represents a control action of the rule R_i of B_j

$$R_i : \text{IF } Q_i, \text{ THEN } A_i, \quad i = 1, \ldots, n,$$

where n is the number of rules of B_j. For example $B_1(x,y)$, is the action suggested by the first behavior in the state x.

Our robot is a Nomad 200 mobile robot endowed with a ring of 16 sonars and 16 infrared sensors. Infrared sensors are used to detect low height obstacles in short-range distances. In order to compute the input variables, first the sonar and infrared measures are preprocessed to avoid any noisy input data. Then, the sixteen measures from the ring of sensors are used as the distance in inches to the detected objects. The information provided by the sensors or the world model (depending on the kind of behavior) is fuzzified in several input variables that are used by the behaviors to compute their preferences among the control actions.

The two control variables that have been considered are: steering velocity in $\frac{1}{10}$ of degrees per second (Fig. 1) and translation velocity or speed in $\frac{1}{10}$ of inches per second (Fig. 2). Speed is increased in Δspeed (Fig. 3), if the robot is reaching the objective of control and is reduced whenever it is not. Each behavior therefore has two rule bases: one to control the steering velocity and another to control the translation velocity.

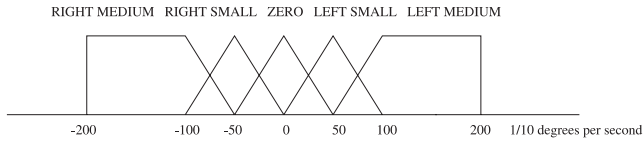

Fig. 1. Steering velocity.

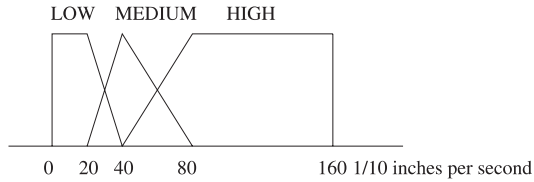

Fig. 2. Translation velocity or speed.

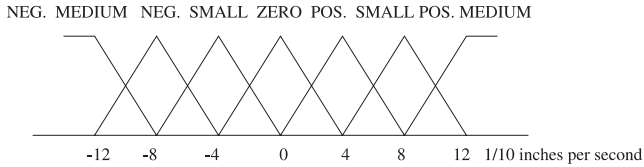

Fig. 3. Δspeed.

3.1. Design of objective-oriented behaviors

The design of a behavior is actually the design of fuzzy rules set and the definition of the shape of the linguistic variables membership functions of the fuzzy sets collected in the rules. If the behavior belongs to the first kind, i.e., it is an *Objective-oriented* behavior, then the rule base can be defined by means of a rule base typical of regulatory control. It is necessary to define a set-point or objective, for example if the objective is to follow the right wall to a certain distance and the robot has to be aligned to the wall, then the input variables will be the current distance to the right wall (right distance or RD, Fig. 4) and

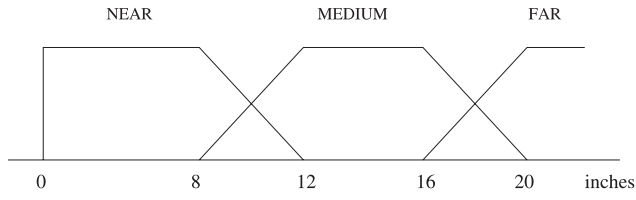

Fig. 4. Right or left distance.

the angle to it (angle to right wall or *AngRW*, Fig. 5). Thus, the set-point will be defined using a rule $R_{\text{objective}}$ as follows:

To follow the right wall and to control the steering velocity

$$R_{\text{objective}}(\textit{Follow wall}) = \text{If right distance is MEDIUM and}$$
$$\text{angle to right wall is ZERO, then steering velocity is ZERO,}$$

where MEDIUM, and ZERO are linguistic labels for each variable, respectively.

The entire rule bases and the linguistic labels of the variables for Follow wall are described here below:

(a) For the steering velocity control which depends on RD (rows) and AngRW (columns), the rule base is described in Table 1.

(b) For the translation velocity (speed) control, the rules are:

If speed is {MEDIUM or LOW} and RD is MEDIUM and AngRW is ZERO, then Δspeed is PS.

If speed is HIGH and RD is {FAR or NEAR}, then Δspeed is NS.

If speed is HIGH and AngRW is {POS or NEG}, then Δspeed is NS,where Δspeed is the speed variation in $\frac{1}{10}$ of inches.

Other behaviors of this kind are:

3.2. Follow corridor behavior

The objective of the control of the *Follow corridor* behavior is to keep the robot close to the middle of a corridor and to keep it in line with it. In order to do so, the input variables considered are, respectively, the difference between the right distance and left distance (RD − LD) and the difference between the

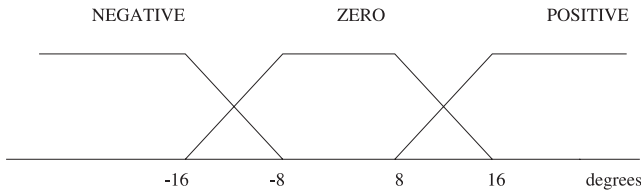

Fig. 5. Angle to right or left wall.

Table 1
Rules to control the steering velocity in *Follow wall* behavior

	NEG	ZE	POS
FAR	RM	RS	ZE
MEDIUM	RS	ZE	LS
NEAR	ZE	LS	LM

angle to right wall and angle to left wall (AngRW – AngLW), thus the first variable is used to keep the robot close to the middle and the second one to keep it in line with the corridor. For this and the following behavior only the $R_{\text{objective}}$ is described:

$R_{\text{objective}}$(Follow corridor)
= If RD – LD is ZERO and AngRW – AngLW is ZERO,
then steering velocity is ZERO.

3.3. Face object behavior

This behavior moves the robot according to a certain orientation so that the robot aligns itself with the corresponding object (wall, corridor or door). This behavior is very useful for aligning the robot with a door since the *Cross door* behavior implemented requires the robot to be facing the door to be crossed. Also, if the robot is facing a wall or a corridor then it is easier to detect the beginning of the object. This behavior uses the difference between the current orientation and the goal orientation (CO – GO) as an input variable to control the steering velocity and its $R_{\text{objective}}$ is:

$R_{\text{objective}}$(Face object)
= If CO – GO is ZERO, then steering velocity is ZERO.

3.4. Design of a typical purely reactive behavior: avoid obstacle

The *Avoid obstacle* behavior has been designed to avoid an unexpected obstacle wherever the robot goes. It means that the robot can be, for example, following the left wall, the right wall or in the middle of a corridor when a frontal obstacle is detected. This behavior uses the right and left distance (RD and LD, respectively, Fig. 4), the frontal distance (FD, Fig. 6) and the angle to the obstacle (AngO, Fig. 7) as input variables to avoid the unexpected frontal obstacle. The behavior to avoid obstacles should take the preference of the other behaviors into account so that the robot can avoid the obstacle while it is, for example, following the wall on the right, although there will be many

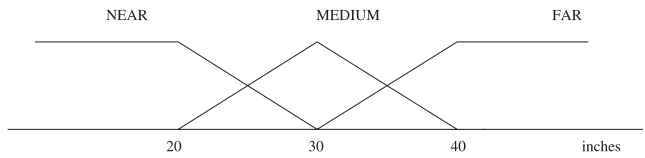

Fig. 6. Frontal distance.

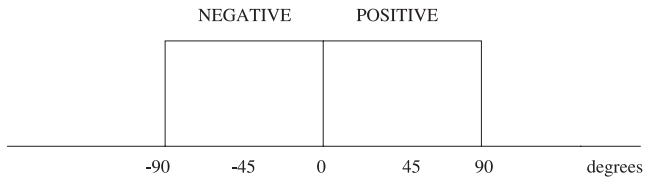

Fig. 7. Angle to obstacle.

situations in which this trade-off is not a possible solution. In order to allow the best trade-off among several preferences, the behavior *Avoid obstacle* is based on two different policies. During the first policy, the robot avoids obstacles according to the right distance and the left distance, moving to the more distant side. But if the right distance and left distance are large enough then the *Avoid obstacle* behavior has no preference when deciding which way to turn. At this moment, the other behavior could choose the way to turn according to its own preference. For example, if the other behavior is *Follow wall* and the robot was approaching the right wall when the obstacle was detected then this behavior can choose to turn to the right in order to approach the wall and avoid the obstacle at the same time (Navigation Task 3 in Section 7 shows this example). The second policy is activated by the reaching of a certain frontal distance to the obstacle: for example if the robot is following a corridor, the *Follow corridor* behavior is attempting to keep the robot close to the middle, and if the obstacle is in front, then the *Avoid obstacle* behavior has to choose which way to turn and it will do so according to the angle to the obstacle so that the robot will turn to the best side. The *Avoid obstacle* behavior will only be combined with the other behaviors during the first policy. Of course, the transition between the first and the second policy is smooth because the transition depends on the linguistic variable frontal distance. In all cases the speed is reduced to move the robot slowly when an obstacle is detected in its vicinity.

The rule bases for avoid obstacle are described here below:

(a) The rule base for the steering velocity control which depends on RD (rows), LD (columns) during the first phase, in which DF is MEDIUM, is shown in Table 2, where H is the convex hull of the labels RS, ZERO and LS.

Table 2
Rules to control the steering velocity in *Avoid obstacle* behavior (FD is medium)

	NEAR	MEDIUM	FAR
NEAR	H	LS	LS
MEDIUM	RS	H	LS
FAR	RS	RS	H

In the second phase the frontal distance is NEAR. In this case, the steering velocity control depends on the angle to obstacle (AngO). The rules are:

If FD is NEAR and AngO is POS, then SV is RM,
If FD is NEAR and AngO is NEG, then SV is LM.

(b) For the translation velocity (speed) control in both phases, the rules are:

If SPEED is HIGH and FD is {MEDIUM or NEAR}, then ΔSPEED is NM
If SPEED is MEDIUM and FD is NEAR, then ΔSPEED is NS.

3.5. Design of short-memory behaviors

These behaviors need more information about the environment because the sensory information can be very inaccurate, thus a partial model of the world is constructed gathering sensor data and filtering the noisy data. For example, in order to *Cross a door*, the free space between both sides of the door-frame is usually very small and the sonars and infrared can provide very inaccurate measurements. So a model of the door-frame is constructed and a trajectory through the middle of the space between both sides is generated. The robot must follow this trajectory instead of taking the sensor data as input information for the behavior. If the trajectory becomes dangerous or the robot is too near a door-frame, a new model for the door-frame is constructed using the information from the sensors and a new trajectory is generated. With this method for designing these kinds of behaviors, we separate the fact of generating the trajectory from the fact of following that trajectory. The following of the trajectory is resolved by means of a rule base that considers the distance and angle to the trajectory as state variables so that the rule base is very similar to the rule base of an *Objective-oriented* behavior. This is important since it allows us to obtain a good complex behavior when there is a transition from another behavior to this one.

In any case, while the robot is going through a door the distances on the right and left are very small, and it therefore has to be very careful so as not to crash into the door-frame, thus the speed is reduced so that the robot moves slowly.

4. Methods for behavior combination

Two methods are used to combine the basic behaviors: the first is based on the context-dependent blending proposed in [27] and the second uses a linear

combination of the defuzzified behavioral response [22]. This second method can be shown to be equivalent to a vector summation scheme. This scheme is a very popular approach for combining behaviors in which each command is represented by a force vector and commands from different behaviors are combined by vector summation. Other proposals [13,24] that use fuzzy logic to perform command fusion actually implement a trivialized form of fuzzy command fusion which can be shown to be equivalent to a vector summation scheme [25].

Fuzzy metarules are used to select the appropriate application context for a particular behavior ensuring smooth transitions between states, so that the complex observable behavior is implemented by means of the collaboration of simpler behaviors. The activation of one of these behaviors is then determined by the robot-environment states using the truth value of the context of applicability.

The influence of the context of applicability is computed differently according to the combination method used. We call the first method C&D (combine and then defuzzify) and the second method D&C (defuzzify and then combine). In both methods, the metarules give us the activation values of the behaviors according to the current context:

$$R_k : \text{IF } C_k, \text{ THEN } B_k, \quad k = 1, \ldots, m,$$

where C_k is a fuzzy proposition that determines the applicability of the context in the state x and m is the number of behaviors that will take part in the navigation task. In C&D the influence of the context is obtained using Eq. (2)

$$B_j^C(x, y) = \{C_j(x) \Rightarrow B_j(x, y)\}, \tag{2}$$

where $\Rightarrow$ is a fuzzy implication operator. For example, the Gödel implication Eq. (3) can be used

$$a \Rightarrow b = \begin{cases} 1 & \text{if } a \leqslant b, \\ b & \text{otherwise.} \end{cases} \tag{3}$$

Therefore the fuzzy set final value FV (Eq. (4)) is obtained by Eq. (4)

$$FV(x, y) = B_1^{C_1}(x, y) \cap B_2^{C_2}(x, y) \cap, \ldots, \cap B_m^{C_m}(x, y), \tag{4}$$

where x is the input array that defines the actual state, y represents the possible values in the output variable and m is the number of behaviors.

Finally, the fuzzy set $FV(x, y)$ must be defuzzified, for example by using the center of gravity method.

In the second method, D&C, first each $B_j(x, y)$ is defuzzified by the center of gravity (Eq. (5)):

$$Df_j(x) = \frac{\int yB_j(x,y)\,\mathrm{d}y}{\int B_j(x,y)\,\mathrm{d}y} \tag{5}$$

obtaining a numerical value for each activated behavior. The composition is computed, weighting each of these values times the degree of applicability of C_j. Let $\lambda_j = C_j(x)$ be the degree of applicability of C_j, a value between 0 and 1 that represents the applicability of C_j in the state x. Then the final value in this case ($DFV(x)$), is given by Eq. (6):

$$DFV(x) = \lambda_1 Df_1(x) + \lambda_2 Df_2(x) + \cdots + \lambda_m Df_m(x). \tag{6}$$

In many cases, both methods will give similar results but below, we consider a typical situation, where both methods would give different final outputs. One behavior needs to approach the wall, and an obstacle has been detected. Let us suppose that the applicability of the first behavior is 0.5 and the applicability of obstacle avoidance is 0.5, then the two methods of combination would give the following outputs: C&D gives a final output for the steering velocity of $50\frac{1}{10}$ degrees per second while D&C gives a final output of $25\frac{1}{10}$ degrees per second. This is shown in Fig. 8.

This example shows different final outputs with the same input data. One idea of the experiments is to find out if this difference is important in the complex fused behavior and to see if it can influence the final trajectory of the robot. Several examples of the influence of the combination method are shown in Section 7.

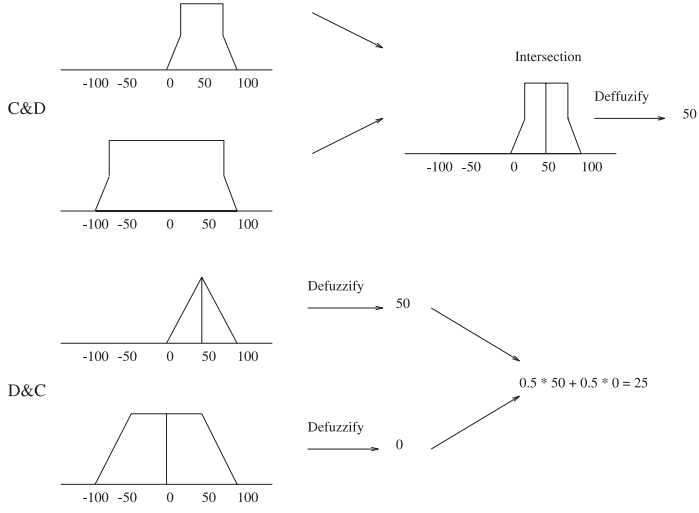

Fig. 8. The final output may be different.

5. Contexts description

In order to decide which behavior or behaviors should be activated at each moment, a set of fuzzy metarules is generated by the executive level from the plan that has been computed by the planning level. To compute the plan, the robot needs to know and to manage some information about the environment. This information will be represented by means of a topological map, which is constructed by performing a previous exploratory task and discovering several kinds of distinctive places. These places are regions in the world that have characteristics that distinguish them from their surroundings, such as corners, walls, corridors or doors. The topological map formally consists of a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of N nodes, and $E = \{e_{ij}\}$ with $i = 1, \ldots N,\ j = 1, \ldots, N$, where $e_{ij} = (v_i, v_j)$, is the set of M edges. The nodes can be classified in one of the following kinds of detectable landmarks: corners, doors, hallways and a default landmark type corresponding to long irregular boundaries. On the other hand, the arcs are the walls or corridors that connect the nodes. Both nodes and arcs have a landmark descriptor which contains information about the landmark type, and if necessary, a fuzzy estimate of the landmark's length.

This topological map provides information directly gleaned from the robot's experiences within the world, but it can additionally contain any information obtained independently from the robotic agent itself, such as maps obtained from floorplans. This kind of information resolves, for example, the problem of detecting the presence of a staircase. The topological map can be used by the planner to compute a minimum-cost path from the current position to the desired goal using the estimated length of each arc using a standard graph search algorithm, such as Dijkstra's shortest path algorithm or A^* algorithm. The robot, by means of the activation of its reactive behaviors, will then be able to accomplish the plan in spite of the presence of unexpected obstacles along its path. For example, the plan computed to move the robot from a corridor to a corner at the end of a wall (the goal), may be defined as the robot having to follow the corridor *CorridorX* and after that following the wall *WallY* on the right until the final goal is reached, of course avoiding the possible obstacles. The translation to metarules is carried out by the executive level directly, coupling landmarks to behaviors. Thus the set of metarules will be as follows:

If *obstacle* then *Avoid obstacle*

If $\neg$*obstacle* $\wedge$ In(*CorridorX*) $\wedge$ $\neg$ In(*WallY*)
then *Follow corridor*(*CorridorX*)

If $\neg$*obstacle* $\wedge$ In(*WallY*) $\wedge$ $\neg$ In(*Goal*)
then *Follow wall*(*WallY*)

If $\neg$*obstacle* $\wedge$ In(*Goal*) then *Stop*.

The last metarule is just to indicate the stop condition since *Stop* is not actually a behavior. In this example the contexts are given either by the presence or absence of an obstacle and by the detection of the corridor *CorridorX* or the wall *WallY*. The detection of obstacles, corridors and walls is based on data from the sensors which are processed by several perceptual routines. These routines infer the existence of the obstacle, wall or corridor and afterwards this perception is matched with the topological map of the environment so that the robot can recognize the place where it is.

In our proposal, reasoning is based on the path planning process because the relation between the landmarks and behaviors is natural and direct. This allows the robot to compute the best path in order to achieve the final goal, although the environment may have several paths joining the current position and the goal. This question is an important difference between our system and the work of Saffiotti [27], since it is not as easy to optimize the cost of the computed path in the behavioral plans as it is in our proposal.

With these metarules different behaviors can be activated at the same time but each behavior only makes a contribution if its own context of applicability is true to some degree. For example, an extreme case would be the activation of *Avoid obstacle*, *Follow corridor* and *Follow wall* at the same time. The final output will be the result of combining each behavior output previously modified by the truth value of its context of applicability.

An important aspect of behavior fusion is that if the combination method is based on the intersection of each behavior output then the possibility of conflict among preferences of behaviors should be taken into account. The possibility of an empty intersection between the preferences of the behaviors may result in the selection of an undesirable control value in both combination methods. An empty intersection can be produced when one behavior prefers to turn to the right for steering velocity and the other prefers to turn to the left. In this case, the first method (C&D) should avoid this situation or know how to deal with it in order to choose a value. The second method (D&C) could blend every defuzzified value but the final value would instruct the robot to go straight on and this control value is not preferred by any of the behaviors and could lead the robot into dangerous situations. The fact that the defuzzification may result in the selection of an undesirable control value, has already been addressed in the literature in this field. In Yen and Pfluger [29] the authors define a new defuzzification method to solve a similar problem, others like Saffiotti et al. [27] place the responsibility on the behavior designer to make sure that rules with conflicting consequents have disjointed antecedents, and others like Goodridge et al. [12] allow the control system to fail safely and recover from the failure by means of a high-level processing that is responsible for analyzing and resolving the problem. In this work this aspect is dealt with in the following way. First, not every combination makes sense. In our case the following of a wall could be on the right or on the left but both behaviors will

not be activated at the same time. The designer should decide which behaviors can be combined at the same time and how to deal with the possible conflicts among behaviors. One possible solution is simply not to allow these conflicts. This would mean that when the environmental state is the same, the combined behaviors will never have disjointed outputs. To do this, it would be necessary to know the states in which several behaviors can be combined and determine the values for the input variables from each behavior in such a state. The designer could therefore avoid the occurrence of disjointed consequents in the rules. This solution has, in our opinion, two main disadvantages: first, the design of the rule base of a behavior is influenced by other external aspects of this behavior and therefore some advantages such as the simplicity of the design would be lost; and second, the difficulty of determining the consequents from each behavior in a certain environmental state because the input variables of each behavior could be very different.

Our solution is to relax the condition of consistency and to permit the existence of disjointed consequents in some cases. That is to say, the intersection among the preferences of behaviors may be empty. In such cases, the output from the behavior with the highest truth value in its context would be taken as the final output. If the truth values are equal, then the most recent behavior would have priority over the older ones. The advantage of this solution is that the generation of the metarules is simpler because they do not have to consider any problem relative to the intersection of the consequents from every behavior, thus the generation of the metarules only has to consider aspects relative to the context perception and the accomplishment of the plan. Another advantage is that there is a higher interaction among behaviors and the fact that if there is a conflict it would be resolved by giving preference to one of them but in running time instead of design time. In our implementation we have chosen to develop this latter solution to avoid the problem of the possible empty intersection of some consequents and this is the solution that we have used in both combination methods.

6. Evaluation measures

The objective of the experimentation is to test the architecture performance, the quality of the control and the smoothness and the length of the robot trajectory. The quality of the control is related to the achievement of the control objective in each behavior since this objective represents the set-point of the underlying controller to the behavior. The measures related to the achievement of the control objective are interesting because:

- These measures allow us to test the goodness of the rule base and the definition of the membership functions of the fuzzy sets collected in the rules, in order to achieve the proposed objective of each behavior. For example, if the

behavior *Follow wall* has a high value of quality of control, then this means that the behavior is able to maintain the robot near the wall and in line with it, then the sonar data will be very accurate since the value of the distance to the wall is computed by means of the echo of a sonar beam perpendicular to that wall. This sonar data could be used, for example, to build a map of the environment.

- Moreover, the measures related to the achievement of the control objective can be used to tune the labels of the fuzzy sets of the behaviors, using some machine learning algorithm.

On the other hand, measuring the smoothness and the length of the robot trajectory will give us an idea of the energy that the robot can consume during its motion. Also, if the robot has to turn to one side, a smooth motion can produce less odometry error than a tight turn according to our experimental observations.

Additionally, the two combination methods have been tested in simulations and in the real world in order to determine the influence of each combination method on the complex behavior. In this section, we will explain the evaluation measures which are used in the experimental part. The idea is to use two kinds of measures: one to determine the achievement of the control objective and the other to measure the smoothness of the robot trajectory.

6.1. The achievement of the control objective

When a fuzzy controller has a unique rule base for controlling certain variables such as the distance and angle to the wall, it is easy to test its operation by computing the error from these variables. In fact, each behavior was tested using this kind of test both in a simulation and also in a real office-like environment before they were combined. However, if we are looking for an evaluation measure for following the wall and the corridor, then the error from the input variables in each behavior should be computed and fused by using a normalization method. Another means of evaluation is to take the truth value of the rule which defines the control objective or set-point. These measures will be defined, first by taking into account only one behavior and afterwards by taking into account several behaviors at the same time. According to the methodology of design previously explained, every behavior, with the exception of the purely reactive behavior *Avoid obstacle*, has been designed to achieve and maintain some control objective such as following a wall at a certain distance, or following the center of a corridor, or crossing a door following a trajectory, or achieving a certain orientation. In these behaviors, the rules must guide the robot towards a specific situation defined using certain values of the input variables. However, in the case of the behavior *Avoid obstacle*, the situation to be achieved is simply not to crash into the obstacle and this fact can guide the robot to multiple and indeterminate situations, which

are not actually defined. Therefore, the behavior *Avoid obstacle* has not been considered like a behavior of this kind.

With B_j being such a behavior, there must be a rule, relative to the steering velocity control, that defines the control objective. This rule, as we said above, is called $R_{\text{objective}}(B_j)$.

These rules follow the general expression shown below:

$$R_1 = \text{If } X_1 \text{ is } A_1 \text{ and } X_2 \text{ is } A_2 \text{ and}, \ldots, \text{and } X_n \text{ is } A_n, \text{ then } Y \text{ is } D_j,$$

where A_i, with $i = 1, \ldots, n$, are linguistic labels of the n antecedent variables X_i, $i = 1, \ldots, n$, respectively and D_j is a linguistic label of the consequent variable Y. The truth value of one rule R_1 in the state x, where $x \equiv \{x_1, x_2, \ldots, x_n\}$ is the input array that defines the current state, is given by Eq. (7)

$$TV(R_1, x) = \min_{1 \leqslant i \leqslant n} (\mu_{A_i}(x_i)), \tag{7}$$

where $\mu_{A_i}(x_i)$ is the membership value of x_i to the linguistic label A_i.

The truth value of $R_{\text{objective}}$ represents the achievement of the control objective such as following the wall or corridor. Therefore, whenever the behavior is activated its truth value is taken into account for the evaluation of the final complex behavior. There are two evaluation measures related to the truth value of $R_{\text{objective}}$.

(a) *Global achievement*

$$G_a(B_j) = \frac{1}{N} \sum_{k=1}^{N} TV(R_{\text{objective}}(B_j), x_k), \tag{8}$$

where N is the total number of control cycles in which the behavior B_j has been activated, i.e., when the value of its context is greater than zero, and $TV(R_{\text{objective}}(B_j), x_k)$ is the truth value of $R_{\text{objective}}$ of B_j in the state x_k in which the behavior B_j has been activated.

(b) *Restricted achievement*

$$R_a(B_j) = \frac{1}{M} \sum_{k=1}^{M} TV(R_{\text{objective}}(B_j), x_k), \tag{9}$$

where M is the number of control cycles, where the $TV(R_{\text{objective}}(B_j), x_k)$ is greater than zero, that is to say, in this case x_k is only representing the states in which the objective has been achieved.

The first measure represents the level of achievement of the control objective by taking into account the environment-robot situations from the first one to the last one while the behavior is activated. However, the second one is the level of achievement of the control objective which only takes into account those situations, where the objective has been achieved. It may therefore be considered as the quality of the control. The range of these measures are values

between 0 and 1 because the maximum value will be from the achievement of the control objective with truth value equal to 1 in every control cycle. If both measures are equal, it means that the robot has reached the control objective at all times, for example following the wall at the correct distance in every control cycle. However, the robot usually loses the control objective when it avoids an obstacle and some time is needed to reestablish the control objective. For example, if the robot is following the wall at the indicated distance and some obstacle is detected in front of it, the robot has to leave the wall it is following to avoid the obstacle and hence it loses the control objective for some time. In this case, the first measure will show the influence of the time needed to reestablish the control objective, whereas the second one will not be affected. Note that the first measure also considers the goodness of the rest of the rules besides the $R_{\text{objective}}$ since if the set-point is achieved soon the measure will be higher.

There are several behaviors which can be activated simultaneously. According to certain metarules, two behaviors can be activated at the same time, for example *Follow corridor* and *Follow wall* can be activated at the same time when the end of the corridor is being reached and the next wall is beginning to be perceived. Therefore, it is necessary to define the truth value of the objective rule in this case: Let B_1 and B_2 be two behaviors and C_1 and C_2 be the truth values of contexts of applicability, respectively, then the truth value of the objective rule in the state x_k is defined by the following equation:

$$TV(R_{\text{objective}}(B_1, B_2), x_k) = \frac{C_1}{C_1 + C_2} TV(R_{\text{objective}}(B_1), x_k) + \frac{C_2}{C_1 + C_2} TV(R_{\text{objective}}(B_2), x_k).$$

The term $(C_1/(C_1 + C_2))$ is needed to weight the truth value of $R_{\text{objective}}$ of each behavior by the weight of its context in the current situation. This makes sense because a behavior has been designed to achieve its objective within a particular context.

Both measures (G_a, R_a) will be used in Section 7 to show the performance of the architecture, the quality of the control and the influence of the combination method on the achievement of the control objective. G_a measures the skill of the system to achieve the objectives from every behavior, taking into account the periods of transition among behaviors, while R_a shows the accuracy and the quality of the control in the objective situation.

The benefits of these measures defined here are as follows:

- In order to compute the measurements, the only condition necessary for the objective of every behavior is that it can be expressed by means of a rule called $R_{\text{objective}}$. This rule is an expression that uses the input variables of the behavior to define its objective. In our architecture every behavior, except *Avoid obstacle*, has its $R_{\text{objective}}$.

- During a navigation task, several behaviors are going to be sequentially activated until the final goal is achieved. Thus, the control objective of the robot changes according to the current behavior. This objective can be to follow the wall, the corridor, etc. The definition of the measures uses the $R_{\text{objective}}$ of each activated behavior and it is therefore possible to measure the global performance of the robot.
- The fact of weighting the truth value of the $R_{\text{objective}}$ of each behavior by the weight of its context, makes it possible to take into account the activation of more than one behavior at the same time.

6.2. The bending energy

The smoothness of the robot motion is an interesting parameter to be considered because the robot can save energy and time while it is moving if the trajectory is smooth. In order to evaluate the smoothness of the robot motion, the curvature of the trajectory will be computed. In the real Euclidean plane, curvature is defined as the rate of slope change as a function of the arc length. For the curve $y = f(x)$, this can be expressed in terms of derivatives as

$$\frac{(\mathrm{d}^2y/\mathrm{d}x^2)}{(1+(\mathrm{d}y/\mathrm{d}x)^2)^{3/2}}. \tag{10}$$

However, when considered digitally, it is not clear how to define an equivalent slope measure. This problem may be overcome by using curvature approximation methods. The method used in this paper is based on curvature approximation by means of orthogonal regression [15].

Once the curvature on every point of the robot trajectory has been computed, it is possible to compute the BE. The BE [28, pp. 202–205], may be understood as the energy necessary to bend a rod to the desired shape, and can be computed as the sum of squares of the curvature at every point of the line $c(x_i, y_i)$ over the length of the line L. Then the BE of a robot trajectory is

$$\mathrm{BE} = \frac{1}{L}\sum_{i=1}^{L} c^2(x_i, y_i), \tag{11}$$

where $c(x_i, y_i)$ is the curvature at every point of the robot trajectory and L is the number of points on the robot trajectory.

The BE defined in Eq. (11) may be used to evaluate the smoothness of the robot motion [1] but such a measurement is an average and it does not show with sufficient clarity the fact that some trajectories will be larger than others. Thus, the measurement that will be used is the TBE as defined by Eq. (12)

$$\mathrm{TBE} = \sum_{i=1}^{L} c^2(x_i, y_i). \tag{12}$$

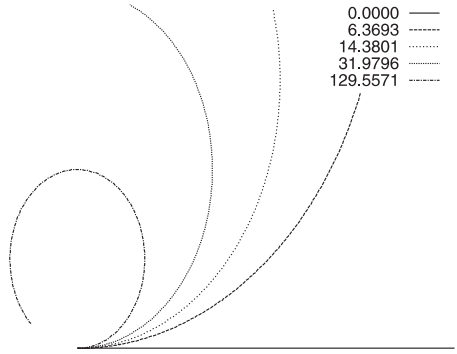

Fig. 9. The TBE.

The TBE is a measurement that takes the smoothness and the length of the trajectory into account simultaneously. This measurement computes the energy expenditure associated with a trajectory so that the trajectories with sharp turns and long length will use a lot of energy and the value of TBE will be high.

Fig. 9 shows the relationship between the TBE and the smoothness of some trajectories. The trajectory with low TBE is the straight line and the energy expenditure increases according to the curvature of the following trajectories.

7. Experimental results

In order to test the performance of the system in both simulation and the real world we have carried out different tasks. The first one was to activate each of the behaviors separately in its own context of applicability in order to tune the linguistic variables of the rule base of each behavior. This tuning process has been based on expert knowledge gained from the results of different trials. After that, several navigation tasks which required different behaviors to be activated were carried out in order to test the performance of the whole system. Both combination methods have been explored with the implemented elementary behaviors, in simulation and in real office-like environments obtaining the different results which are described in this section. Additionally, several speed values have been explored by changing the support of the linguistic labels at the speed domain and good results were achieved, although when the speed is increased it is necessary to modify the distance used in order to activate the *Avoid obstacle* behavior.

At the end of this tuning process, the robot reaches the control objective in every case in spite of the presence of noise in the sensor data and the different surfaces of the walls. When several behaviors are activated, the transition

between behaviors is smooth and the robot adapts its behavior according to the current context.

Before explaining the study of the influence of each combination method, we will detail the simulation parameters used. The sonar sensor has been simulated taking into account the following parameters:

- *Angular range of the main lobe of the sonar*. A cone of 30° is used to simulate the sound beam, and the incidence angle of the beam to the surfaces is considered.
- *Maximum incidence angle*. If the angular difference between the sonar axis and the normal to the surface is larger that 60°, then the simulated sonar is assumed not to have any echo back and the maximum distance will be returned.
- *Random error*. This is an absolute random error factor. It is expressed as a percentage of the real value. The error factor considered was 1%.

Regarding infrared sensors, the most important parameters are listed below:

- *Angular range of the cone*. In this case the cone has a width of 20°.
- *Incidence error*. This is an error factor used with the conical model of the beam. It is expressed as a percentage of the real value which also considers the incidence angle. The resulting distance is computed as, value $\times$ (1+ $\phi \times$ incidence error), ϕ being the angle between the sensor axis and the normal to the surface and incidence error being the value 0.05.
- *Random error*. This parameter has the same meaning as in the case of the sonar sensors. The value was also 1%.

Even though other parameters are used in the simulation of the sonar and infrared sensors, we only show the most important ones here. The remaining parameters can be found in the user's manual for the Nomad 200 robot.

The study of the influence of each combination method has first been carried out by taking into account several navigation tasks in a simulated office-like environment and then, experiments have been carried out in the real world in order to validate the results of the simulation. The simulated environments have the advantage of considering exactly the same environmental conditions for each combination method so that the differences between the final complex behavior can be simply because of using one or another combination method.

In every simulated navigation task, different positions of the obstacles have been tested. We have classified three different types of positions of the obstacles in the environment according to the relative position of the obstacle with respect to the walls of the environment. In the cases of Type I, the obstacle is placed far from the wall that the robot will follow while in the cases of Type II, the obstacle is placed near this wall. In the cases of Type III the obstacle is placed at an intermediate distance from the wall between far and near. Figs. 10–12 depict an environment of Type I, Type II and Type III, respectively, which are used for Navigation Task 1.

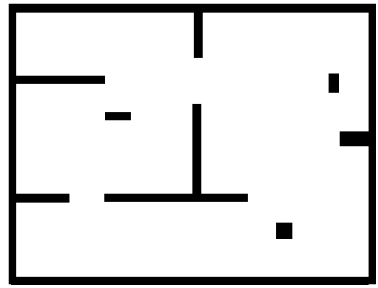

Fig. 10. An environment of Type I.

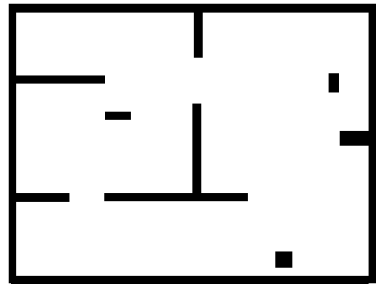

Fig. 11. An environment of Type II.

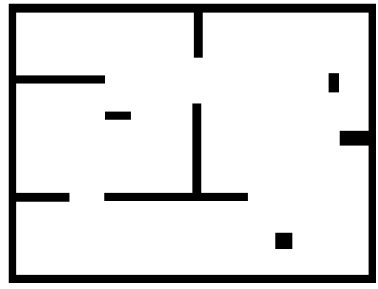

Fig. 12. An environment of Type III.

In order to test both combination methods extensively and compare the influence of the combination method in a given navigation task, we have taken into account in each type of environment 20 different runs of the same navigation task in order to compute the arithmetic average of each evaluation measure. The position of the obstacle in each run has been generated by means

of a random method that produces a position within the range of positions corresponding to that type of environment. Thus, the total number of runs of one navigation task is 60 (20 in each type of environment). In this paper, two navigation tasks in simulation are described so that the total number of experiments is 120 (with each combination method). Notice that different behaviors are activated in these experiments at the same time according to the current context of the robot, thus the combination method plays an important role in the final complex observable behavior.

In each navigation task that we have tested, the results can be summarized as follows. In Type I experiments, the position of the obstacle allows the robot to maintain the control objective with both combination methods, the reason for this being that there is enough space between the obstacle and the object of the environment that is being followed by the robot. On the other hand, in Type II experiments, the position of the obstacle does not allow the robot to maintain the control objective, with either combination method, and it has to follow the boundary of the obstacle. However, in Type III experiments the final complex behavior depends on the combination method that has been used. This means that when the robot uses the C&D method it can maintain the control objective and avoid the obstacle at the same time but when the robot uses the D&C method it cannot maintain the control objective and it has to follow the boundary of the obstacle. In so far as the values of the evaluation measures are concerned, in Type I and Type II experiments, the values of evaluation measures of the C&D method are only slightly better than the values of the D&C method. However, in Type III experiments, the values of evaluation measures of the C&D method are much better than the values of the second one, and thus the experimentation shows that the first method achieves better values of the evaluation measures in the three types of experiments, with higher benefits being obtained in the third type of experiments. Therefore, in the next examples of navigation tasks attention will be focused on Type III experiments, so that a particular run of one experiment of Type III of each navigation task has been selected in order to explain some important points. The description in the first example is highly detailed but in the rest of examples the description is shorter. The proposed navigation tasks are shown below.

7.1. Navigation Task 1

In this navigation task the robot has to reach the goal marked by *Goal*1 in Fig. 13 from an initial position marked by *Wall*1. The plan is computed by the planning level and it is translated to a set of metarules by the executive level. The set of metarules to move the robot from the initial to final position is

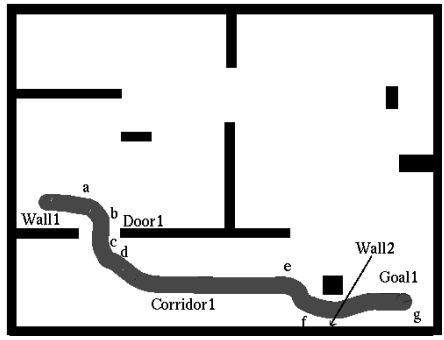

Fig. 13. An experiment of Type III of Navigation Task 1 with C&D.

If *obstacle* then *Avoid obstacle*
If $\neg obstacle \wedge \mathrm{In}(Wall1) \wedge \neg\neg \mathrm{Near}(Door1)$ then $Follow\,wall(Wall1)$
If $\neg obstacle \wedge \mathrm{Near}(Door1) \wedge \neg \mathrm{In}(Door1)$ then $\mathrm{Face}(Door1)$
If $\neg obstacle \wedge \mathrm{In}(Door1) \wedge \neg \mathrm{Near}(Corridor1)$ then $\mathrm{Cross}(Door1)$
If $\neg obstacle \wedge \mathrm{Near}(Corridor1) \wedge \neg \mathrm{In}(Corridor1)$ then $\mathrm{Face}(Corridor1)$
If $\neg obstacle \wedge \mathrm{In}(Corridor1) \wedge \neg \mathrm{In}(Wall2)$ then $Follow\,corridor(Corridor1)$
If $\neg obstacle \wedge \mathrm{In}(Wall2) \wedge \neg \mathrm{In}(Goal1)$ then $Follow\,wall(Wall2)$
If $\mathrm{In}(Goal1)$ then *Stop*.

The metarules give us a natural description of the navigation task, dividing a complex navigation task into a sequence of sub-goals that are achieved by means of the basic behaviors.

Figs. 13 and 14 depict a run of a Type III experiment with C&D and D&C combination methods, respectively, while Figs. 15 and 16 depict the activation level of every basic behavior according to the context of the robot. The sequencing among behaviors emerges from the interaction of the behaviors with the environment, thus a behavior is deactivated when its context becomes false and the transition among them is a smooth transition since the context changes smoothly. The change of context is marked in Figs. 13–16 by the letters 'a' to 'g'.

A detailed description of a run of this navigation task in a Type III environment is given as follows. The robot begins to execute the behaviors according to the truth value of every metarule which depends on the current context of the robot that is interpreted by several perceptual routines. First, the robot follows the wall *Wall*1 until it senses the door *Door*1 (a), then it must face it in order to cross it. Once it is facing the door (b), the robot crosses it and after that (c) it has to face the next object which is the *Corridor*1. Once it is

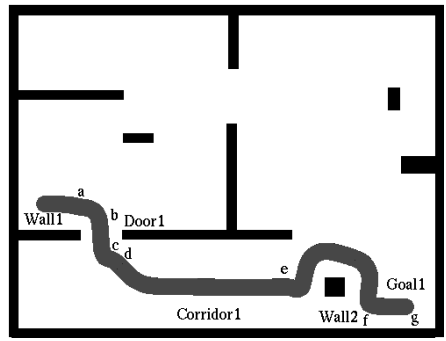

Fig. 14. An experiment of Type III of Navigation Task 1 with D&C.

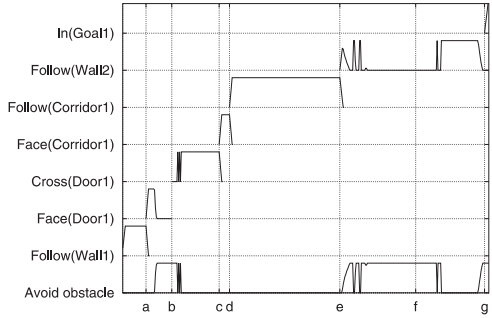

Fig. 15. Levels of activation of each activated behavior during the execution of the experiment with C&D.

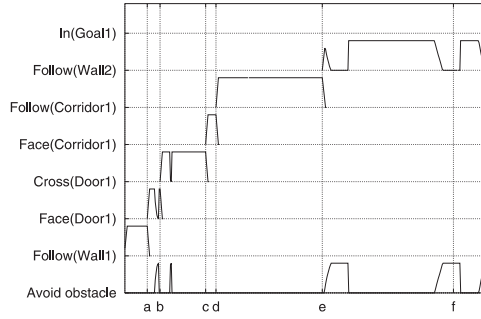

Fig. 16. Levels of activation of each activated behavior during the execution of the experiment with D&C.

facing the corridor (d), it follows the center of the corridor until it arrives at the end of the corridor (e), then it has to follow the wall *Wall*2. At that moment, an unexpected obstacle is detected and activation of *Avoid obstacle* begins, then the *Follow wall* behavior prefers to approach the wall while the *Avoid obstacle* behavior should avoid the obstacle. With the C&D method the combination is dominated by the *Follow wall* behavior because when the robot begins to detect the obstacle the *Avoid obstacle* behavior is indifferent as to which side it should turn to. On the other hand, when the second method is used the robot does not turn sufficiently to the right in order to avoid the obstacle. It therefore approaches the obstacle and the avoidance policy prefers to turn to the left. Label (f) indicates the moment in which the robot begins to get close to the wall *Wall*2. Notice that with the D&C method, the robot begins to get closer to the wall *Wall*2 much later than with the C&D method. Finally, the robot arrives at the goal at the point marked by (g).

As far as the other two types of experiments are concerned, the Type I experiments (obstacle far from the wall) produce trajectories which look like those shown in Fig. 13 while the Type II experiments (obstacle near the wall) produce trajectories which look like those in Fig. 14. In Type I and Type II experiments, the trajectories are quite similar with both combination methods, however, Type III experiments produce the first kind of trajectory when the C&D method is used and the second one when the D&C method is used. Thus, in the Type III experiments, both trajectories are very different.

The arithmetic average of evaluation measures of 20 experiments of each type of Navigation Task 1 are shown in Table 3 and they show that the values of the first combination method are always similar to or better than the results of the second one. The restricted achievement of the control objective is similar in all cases. This means that when the robot has reached the control objective it manages to maintain it very well. However, the global achievement values show that the robot loses the control objective for less time when using the first combination than it does when the second one is used, and this is true in the three types of experiments. Furthermore, the TBE value shows that there is a higher energy expenditure in the trajectories of the second method because they are not as smooth and their lengths are longer than the trajectories of the first one. The Type III experiments are situations where the first method is much

Table 3
Global achievement, restricted achievement and TBE for Navigation Task 1

Type	G_a		R_a		TBE	
	C&D	D&C	C&D	D&C	C&D	D&C
I	0.7561	0.7502	0.8838	0.8820	46.4521	48.8083
II	0.5445	0.5371	0.8764	0.8761	84.9763	88.4451
III	0.7504	0.5280	0.8942	0.8806	52.6481	111.3685

better than the second one and this is because with the C&D method the preference for the *Follow wall* behavior can dominate the final action when detection of the obstacle begins.

7.2. Navigation Task 2

In Navigation Task 2 the robot has to reach the corner marked by *Goal*2 in Fig. 17 from the initial place marked by *Wall*7. Figs. 17 and 18 depict a run of an experiment of Type III of Navigation Task 2 with the C&D and D&C combination methods, respectively.

The arithmetic average of evaluation measures of 20 experiments of each type of Navigation Task 2 are shown in Table 4.

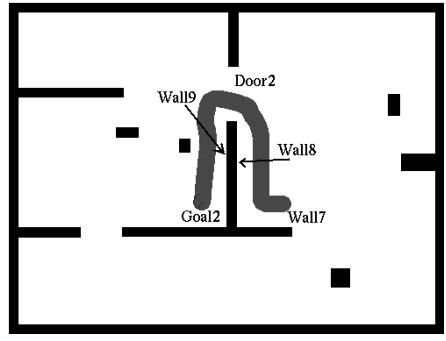

Fig. 17. An experiment of Type III of Navigation Task 2 with C&D.

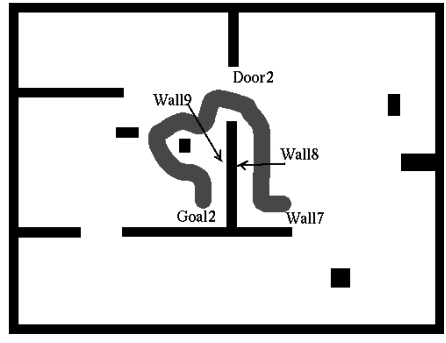

Fig. 18. An experiment of Type III of Navigation Task 2 with D&C.

Table 4
Global achievement, restricted achievement and TBE for Navigation Task 2

Type	G_a		R_a		TBE	
	C&D	D&C	C&D	D&C	C&D	D&C
I	0.5801	0.5443	0.9455	0.9334	55.3207	57.2063
II	0.2592	0.2538	0.9224	0.9213	83.8149	84.1020
III	0.5622	0.2558	0.8744	0.8608	62.6916	123.4410

7.3. Experimentation in the real world

In order to validate the results of simulated experiments, two navigation tasks have been tested in a real world, office-like environment. We have only focused on Type III experiments, since this kind of experiment shows higher differences between the two methods. Every navigation task has been run five times with each combination method and the results are shown below.

7.3.1. Navigation Task 3

In Navigation Task 3 the robot must reach the position marked by *Goal*3 in Fig. 19 from the initial place marked by *Wall*5. Figs. 19 and 20 depict a run of a Type III experiment of Navigation Task 3 with the C&D and D&C combination methods, respectively. In this case, the robot begins to follow the wall *Wall*5 when an obstacle is sensed and avoided. Later, it has to approach *Wall*6 but a new obstacle is situated near the wall. The best action therefore would be to follow the right wall if there is enough space to avoid the obstacle. This trade-off is properly resolved by the C&D method but not by the D&C method.

The arithmetic average of evaluation measures of five experiments of Type III of Navigation Task 3 are shown in Table 5.

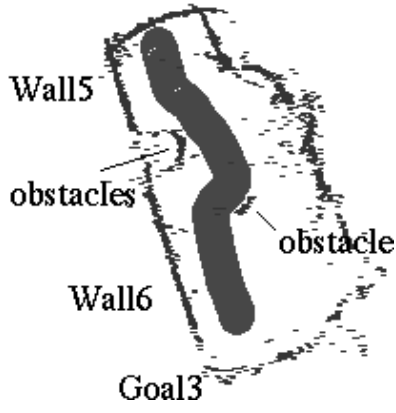

Fig. 19. An experiment of Type III of Navigation Task 3 with C&D.

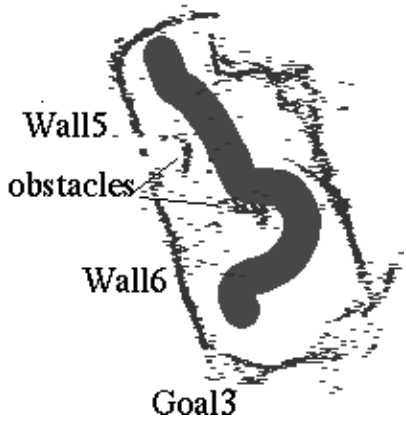

Fig. 20. An experiment of Type III of Navigation Task 3 with D&C.

Table 5
Global achievement, restricted achievement and TBE for Navigation Task 3

Type	G_a		R_a		TBE	
	C&D	D&C	C&D	D&C	C&D	D&C
III	0.4873	0.1334	0.8216	0.8156	42.8932	84.5104

7.3.2. Navigation Task 4

In Navigation Task 4 the robot has to reach the position marked by *Goal*4 in Fig. 21 from the initial place marked by *Co*1. Figs. 21 and 22 depict a run of a Type III experiment of Navigation Task 4 with the C&D and D&C combination methods, respectively. In this experiment, the robot is following the corridor *Co*1 and when the corridor ends, it has to turn the corner and use the behavior *Face wall* in order to be able to detect the left wall and follow it. Near this wall, there is a column that should be avoided. Once again, with the first method the robot can avoid the column and begin to follow the left wall but with the second method this is not possible and the robot must follow the perimeter of the column.

The arithmetic average of evaluation measurements of five Type III experiments of Navigation Task 3 are shown in Table 6.

7.4. Discussion concerning the combination methods

Both combination methods achieve a sufficiently high control performance once the control objective has been reached since the value of R_a is always close to the value 1, which is the maximum. Regarding G_a, the experiments show

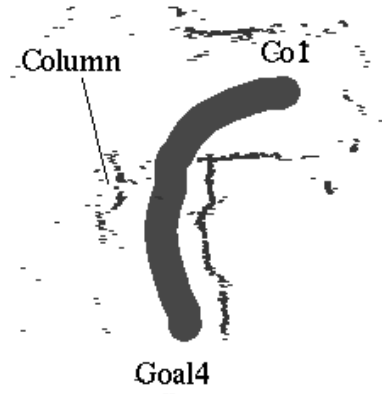

Fig. 21. An experiment of Type III of Navigation Task 3 with C&D.

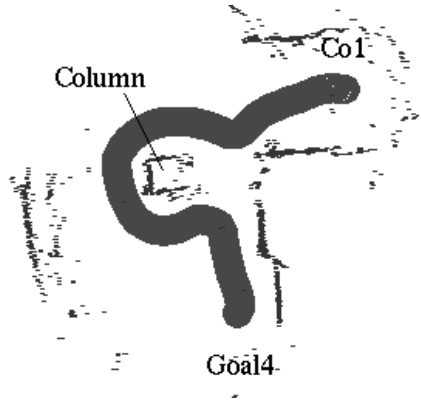

Fig. 22. An experiment of Type III of Navigation Task 3 with D&C.

Table 6
Global achievement, restricted achievement and TBE for Navigation Task 3

Type	G_a		R_a		TBE	
	C&D	D&C	C&D	D&C	C&D	D&C
III	0.4116	0.1720	0.8673	0.8418	16.4082	140.1081

that depending on whether the environment lets the robot maintain the control objective, then G_a will be close to the value R_a. But this does not depend only on the environment since the first method always obtains similar or better values than the second one concerning G_a.

Thus, G_a allows us to assess how the robot achieves the different objectives so that the trajectories which have a low G_a value can be considered worse than those which are associated with a higher value. The differences are higher in the Type III experiments because in these experiments the obstacles are placed in critical positions that do not allow an easy trade-off among the activated behaviors. This trade-off is possible with the first combination method because the preference of a behavior, when the other behaviors are indifferent over the range of possible preferences, is better considered than with the second combination method. This fact has an important influence on the achievement of the control objective because if the composition method can take the preference of one behavior over the composition into account, the achievement of the control objective will be high.

When considering the smoothness of the robot motion, the combination method also has an important influence on the TBE because if the robot loses the control objective it has to expend more energy in order to recover it.

Regarding experimentation in the real world, both combination methods have also been tested in an office-like environment in the real world. The results validate the simulated experiments so that the best values are related to the first combination method.

8. Conclusions

This work addresses the generation of complex behaviors by the combination of simpler behaviors as the lowest level of a hybrid deliberative–reactive architecture for mobile robot navigation. According to the proposed methodology for designing the elementary behaviors, these are implemented by means of fuzzy rules which allow the linguistic descriptions of the control strategies to be easily understood and provide robust navigation capabilities for a mobile robot. Fuzzy logic also offers useful mechanisms to address the arbitration of behaviors and the problem of combination. Arbitration is resolved through the use of fuzzy metarules that are generated by the executive level according to the plan computed by the planning level. The most important features of the architecture proposed here are as follows. The methodology for designing the behaviors, on the one hand, allows us to build the rule base of the behaviors so that the structure of the different behaviors becomes quite homogeneous and enables us to define interesting evaluation measures. On the other hand, this methodology allows us to combine behaviors that use different kinds of input information such as sensor data or the data from sensor-derived world modeling. Regarding the planning algorithms, our system uses minimum-cost path searching algorithms to compute the path between the initial and final position of the robot, and the translation to metarules is carried out in a straight way. If this path becomes

unfeasible, the map of the environment is updated and a new path is computed. With regard to the possibility of conflict among preferences of behaviors, our solution is neither concerned with the design of the rule base nor the design of the metarules and allows a higher interaction among behaviors.

Two combination methods have been studied and tested both in simulation and in the real-world to show their influence on the final complex behavior. The evaluation of the experimentation has been based on three evaluation measures: the global achievement G_a, the restricted achievement R_a and the TBE. The results of our experiments show the good performance of the architecture and the fuzzy behaviors in spite of the presence of noise and vagueness in the sensor data and unexpected obstacles. Regarding the combination methods, the influence of the combination method on the final complex behavior is important because the final trajectories may be quite different in those situations in which a good trade-off among the behaviors is possible. The first combination method produces trajectories that have better global achievement and less TBE in all cases but the benefits are higher when the composition is dominated by the preference of one behavior and there is the possibility of a good trade-off among the behaviors. Therefore, in our system the defuzzification should be performed after combination in order to obtain the best trade-off among the behaviors that are activated at the same time.

In this paper, attention has been focused on the design, coordination and fusion of the different behaviors but needless to say, there are far more issues than we have touched on here. Further work will study learning techniques in order to improve the fuzzy behaviors, where the measures here explained can be very useful for evaluating the robot motion. As far as the rest of the architecture is concerned, we are currently working on the further development of planning algorithms under uncertainty and localization techniques to deal with the vagueness and uncertainty of the real world.

References

[1] E. Aguirre, M. García-Alegre, A. González, A fuzzy safe follow wall behavior fusing simpler fuzzy behaviors, in: Proceedings of the Third IFAC Symposium on Intelligent Autonomous Vehicles, Madrid, Spain, 1998, pp. 607–612.

[2] R.C. Arkin, Motor schema based navigation for a mobile robot, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1987, pp. 264–271.

[3] R.C. Arkin, Motor schema based mobile robot navigation, Int. J. Robot. Res. 8 (4) (1989) 92–112.

[4] R.C. Arkin, Integrating behavioral, perceptual and world knowledge in reactive navigation, Robot. Autonomous Syst. 6 (1990) 105–122.

[5] R.C. Arkin, Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.

[6] B.C. Arrúe, F. Cuesta, R. Braunstingl, A. Ollero, Fuzzy behaviors combination to control a non-holonomic mobile robot using virtual perception memory, in: Proceedings of the IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 1997, pp. 1239–1244.
[7] A. Bonarini, Anytime learning and adaptation of fuzzy logic behaviors, Adaptive Behavior Journal, Special issue on Complete Agent Learning in Complex Environments 5 (3&4) (1997) 281–315.
[8] A. Bonarini, F. Basso, Learning to compose fuzzy behaviors for autonomous agents, Int. J. Approx. Reasoning 17 (4) (1997) 409–432.
[9] R.A. Brooks, A robust layered control system for a mobile robot, IEEE J. Robot. Automation RA 2 (1986) 14–23.
[10] D. Driankov, H. Hellerdoorn, M. Reinfrank, An Introduction to Fuzzy Control, Kluwer Academic Publishers, Dordrecht, 1993.
[11] J. Gasós, M.C. García-Alegre, R. García, Fuzzy strategies for the navigation of autonomous mobile robots, in: Fuzzy Engineering Toward Human Friendly Systems, Proceedings of the International Fuzzy Engineering Symposium (IFES'91), Yokohama, Japan, 1991, pp. 1024–1034.
[12] S.G. Goodridge, M.G. Kay, R.C. Luo, Multi-layered fuzzy behavior fusion for reactive control of an autonomous mobile robot, in: Proceedings of the IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 1997, pp. 579–584.
[13] S.G. Goodridge, R.C. Luo, Fuzzy behavior fusion for reactive control of an autonomous mobile robot:marge, in: Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, CA, 1994, pp. 1622–1627.
[14] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[15] A.P. Knoerr, Global models of natural boundaries: theory and applications, Reports in Pattern Analysis 148, Division of Applied Mathematics, Brown University, 1998.
[16] K. Konolige, K. Myers, The Saphira architecture for autonomous mobile robots, in: Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems, MIT Press, Cambridge, MA, 1998, pp. 211–242 (Chapter 9).
[17] D. Kortenkamp, R.P. Bonasso, R. Murphy, Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems, MIT Press, Cambridge, MA, 1998.
[18] J. Latombe, Robot Motion Planning, Kluver Academic Publishers, Dordrecht, 1991.
[19] D. Leitch, P. Probert, Genetic algorithms for the development of fuzzy controllers for autonomous guided vehicles, in: Proceedings of the Second European Congress on Intelligent Techniques and Soft Computing (EUFIT'94), Aachen, Germany, 1994, pp. 464–469.
[20] W. Li, Fuzzy-logic based reactive behaviour control of an autonomous mobile system in unknown environments, Eng. Appl. Arti. Intell. 7 (5) (1994) 521–531.
[21] D. Lyons, A. Hendriks, Planning for reactive robot behavior, in: Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France, 1992, pp. 2675–2680.
[22] V. Matellán, J.M. Molina, C. Fernández, Fusion of fuzzy behaviors for autonomous robots, in: Proceedings of the Third International Symposium on Intelligent Robotic Systems, Pisa, Italy, 1995.
[23] F. Michaud, Selecting behaviors using fuzzy logic, in: Proceedings of the IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 1997, pp. 585–592.
[24] F.G. Pin, Y. Watanabe, Navigation of mobile robots using a fuzzy behaviorist approach and custom-designed fuzzy inference boards, Robotica 12 (1994) 491–503.
[25] A. Saffiotti, Fuzzy logic in autonomous robotics: behaviour coordination, in: Proceedings of the IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 1997, pp. 573–578.
[26] A. Saffiotti, The uses of fuzzy logic in autonomous robot navigation, Soft Comput. 1 (1997) 180–197.

[27] A. Saffiotti, K. Konolige, E. Ruspini, A multivalued logic approach to integrating planning and control, Arti. Intell. 76 (1995) 481–526.
[28] M. Sonka, V. Hlavac, R. Boyle, Image, Processing, Analysis and Machine Vision, Chapman & Hall, London, 1993.
[29] J. Yen, N. Pfluger, A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation, IEEE Trans. Syst., Man Cybernetics 25 (6) (1995) 971–978.