An architecture to coordinate fuzzy behaviors to control an autonomous robot

Andrea Bonarini [1] Giovanni Invernizzi Thomas Halva Labella Matteo Matteucci

Politecnico di Milano AI and Robotics Project, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, I–20133 Milano, Italy; http://www.elet.polimi.it/

Abstract

We propose an architecture to implement coordination among fuzzy behavior modules for autonomous agents, in real–time tasks. Behavior coordination is obtained by fuzzy conditions and motivations evaluated on information provided by intelligent sensors and a world modeler. We also discuss the compatibility of our architecture with cognitive models. We present the application of this architecture in one of the domains we have faced with it: service and edutainment robotics.

Keywords. *Behavior-based Robotics, Cognitive Robotics, Fuzzy Systems, Robotic Architecture, Agent Architecture.*

1 Introduction

One of the possible approaches to robot control design is the so called behavior–based [1]. In such approach, the robot controller is obtained by the cooperative activity of behavioral modules, each implementing a quite simple mapping from sensorial input to actions. Each module operates on a small subset of the input space to implement a specific behavior; the global behavior comes from the interaction among all these modules. One of the major problems in

Email addresses: bonarini@elet.polimi.it (Andrea Bonarini), invernizzi@elet.polimi.it (Giovanni Invernizzi), halva@libero.it (Thomas Halva Labella), matteucci@elet.polimi.it (Matteo Matteucci).

[1] This research is partially supported by the project "PRASSI", co-funded by the Italian Ministry of University and Scientific and Technological Research (MURST) through the ENEA, and partially by the MURST Project "CERTAMEN".

behavior–based robotics is the design of this interaction, usually pre–defined in terms of inhibitory relationships [1] or vectorial composition of the output of the modules [2],[3],[4]. In this paper, we present a new model to design such interaction, based on fuzzy logic. We introduce two sets of conditions associated to each behavior in order to enable, inhibit and compose them in a non–linear way, compatible with a cognitive model. The *activation conditions* for each behavior module are fuzzy predicates which should be verified in order to activate the corresponding behavior module; we call these predicates *CANDO conditions*. Coordination among behaviors active at the same time is implemented by a different set of predicates which represent *motivations* to actually execute the action proposed by each module (*WANT conditions*). CANDO conditions are intrinsically related to the behavior definition and are used to select the behavior appropriate to the specific situation: if they are not verified, the behavior activation does not make sense. WANT conditions represent the opportunity of activating a behavior in a given context. The context is described in terms of internal state, environmental situation, goals, and interaction with other agents.

This approach to behavior activation is a part of a more general cognitive architecture based on a four–layered cognitive model [5] which considers the abstraction flow in knowledge formation from raw data to theory and models through the formation of symbolic concepts.

We represent symbolic concepts by fuzzy models to face the issue of uncertain and imprecise perception; a fuzzy model implements a classification of the available information and knowledge in terms of fuzzy predicates, which have been demonstrated to be a powerful and robust modeling paradigm [6], [3] close to the designers knowledge models [7]. Other researchers [4], [3], [8] have adopted them to implement control systems in autonomous robots for analogous reasons. Moreover, fuzzy conditions make it possible to compose behaviors through weights dependent on the situation and the motivational state of the agent. We adopt this architecture on different robots involved in a wide variety of tasks, such as: document delivery in an office environment, surveillance, and soccer playing in Robocup [9].

In the next section we introduce the overall cognitive architecture describing the role of our behavior management system *BRIAN* (*Brian Reasonably Implements an Agent Noddle*), which is presented in section 3. In section 4 we introduce edutainment robotics and specifically robotic soccer [9] as one of the possible application environments for our architecture, which is general enough to be applied in any situation where a mobile robot has to operate autonomously. Section 5 is devoted to experimental results showing the effectiveness of our approach to obtain complex behaviors by the fuzzy composition of simple behavioral modules.

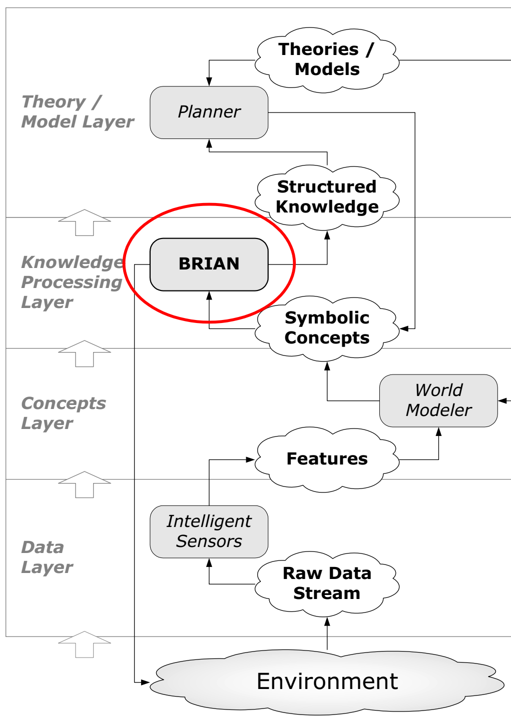


Fig. 1. The overall architecture

2 The overall architecture

BRIAN is the "behavioral" part of a cognitive architecture we use in different applications.

The overall architecture is organized in four layers; each layer represents a different level of abstraction from *raw data* to *theories/models*. Such an architecture is modelled on the reference cognitive model [5](figure 1) we summarize here below:

- *data layer:* extracts basic features from raw data streams as a basis of eventual understanding; this layer is the only agent's information interface to the environment (the other layers work on features extracted by this one)
- *concept layer:* interprets and abstracts features to basic concepts and notions at a higher level of abstraction (*symbolic concepts*)
- *knowledge processing layer:* processes symbolic concepts to obtain more complex information to be used for planning actions, communicating and other reasoning activities
- *theory/models layer:* contains structured abstract models and theories to be used in information processing from "data" to "knowledge"

Intelligent sensors implement the data layer extracting basic features from raw data streams. For instance, omnidirectional vision [10] [11] [12] provides relative positions and distance of colored objects or the the angular position of vertical edges for autonomous robot applications.

A *World Modeler* operates at the second layer to instance, and maintain in time, a conceptual model of the environment inferring symbolic concepts from data features by theoretical models belonging to the 4^{th} layer [13],[14]. In order to do that, basic features extracted from intelligent sensors, are interpreted in terms of higher level features, spatial relations and, then, in terms of ground predicates to generate symbolic concepts in the second layer (e.g., "the red object is close", "the door is on the left"). This process is done according to a cognitive/spatial model which is conceptually part of the fourth layer.

Executive modules in the behavioral component of the architecture belong to *BRIAN*, which uses complex predicates, together with the ground ones, to propose actions for the agent. Complex predicates are built from the ground ones by using logic operators to describe less basic aspects or information at a different level of abstraction.

The *Planner* module reasons on models in the 4^{th} layer to produce goals as symbolic input to *BRIAN*; these concepts are included in the complex predicates used in selecting and blending actions from different behavioral modules. Models in the 4^{th} layer can be abstract maps, graph representations, or even state diagrams, used to forecast or simulate the result of future actions [15].

In this architecture, *BRIAN* implements local (usually reactive) behaviors whose activation is ruled by strategic directions coming from the *Planner*. In other terms, *BRIAN* implements a kind of local reasoning guided by the behavioral plan defined by the *Planner* (*global reasoning*). This, while generating strategic directions, should solve possibly arising conflicts between behavioral modules due to multiple conflicting goals.

3 BRIAN: the behavior management system

BRIAN, our *behavior management system* shown in details in figure 2, uses fuzzy predicates to represent the activation conditions, the motivation conditions and internal knowledge. In *BRIAN* fuzzy predicates may represent aspects of the world, goals, and information coming from other agents. They are represented by a *label* λ, its *truth value* μ_λ, computed by fuzzy evaluation of the input, and a *reliability value* ξ_λ to take into account the quality of the data source. For instance, we may have a predicate represented as

$$< ObstacleInFront, 0.8, 0.9 >$$

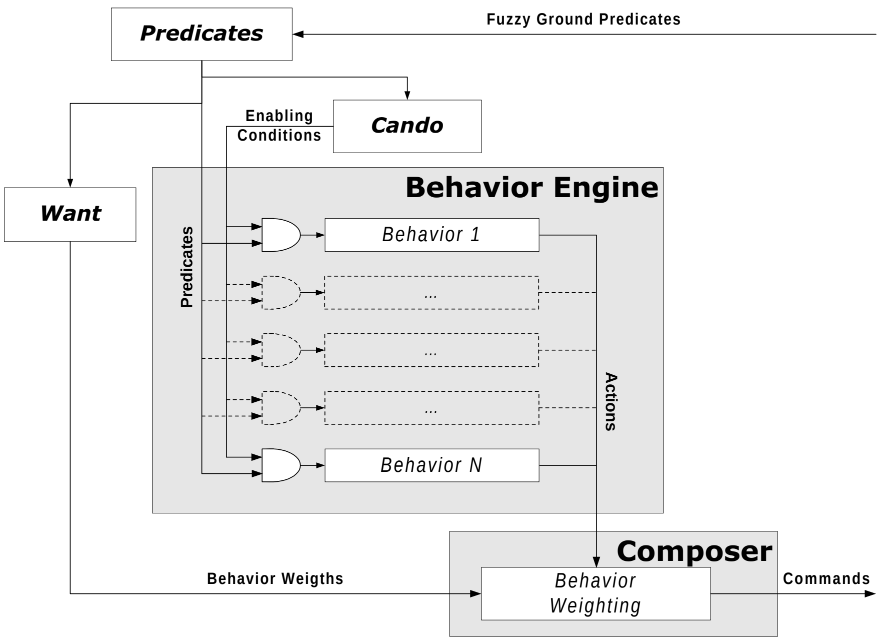


Fig. 2. The behaviour management system

which can be expressed as: “It is quite true ($\mu_\lambda = 0.8$, coming from the fuzzyfication of the incoming real-valued data) that there is an obstacle in front of the robot, and this statement has a reliability quite high ($\xi_\lambda = 0.9$, due to the reliability of the sensing conditions)”

We consider ground and complex fuzzy predicates. *Ground fuzzy predicates* range on data directly available to the agent through the input interface, and have a truth value corresponding to the degree of membership of the incoming data to a labeled fuzzy set. This is equivalent to classify the incoming data into categories defined by the fuzzy sets, and to assign to this classification a weight between 0 and 1. Fuzzy ground predicates are defined on features elaborated by the world modeler and goals from the planner. The reliability of sensorial data is provided by the world modeler basing on perception analysis, and goal reliability is stated by the planner.

A *complex fuzzy predicate* is a composition (obtained by fuzzy logic operators) of fuzzy predicates. Complex fuzzy predicates organize the basic information contained in ground predicates into a more abstract model. In Robocup, for instance, we can model the concept of ball possession by the *OwnBall* predicate, defined by the conjunction of the ground predicates *BallVeryClose* and *BallInFront*, respectively deriving from the fuzzyfication of the perceived ball distance, and the perceived ball direction.

We define for each behavior module a composition of predicates that enable its activation: the *CANDO preconditions*. The designer has to put in this set all the conditions which have to be true, at least to a significant extent, to give sense to the behavior activation. For instance, in order to consider to kick a ball into the opponent goal, the agent should have the ball control, and it should be oriented towards the goal.

Another composition of fuzzy predicates is associated to each behavior module: the *WANT conditions*. These are predicates that represent the motivation for the agent to activate a behavior in relation with a context. They may come either from the environmental context (e.g., *TargetInFront*, which may be expressed as "my target is in front of me"), or from strategic goals (e.g., *CollectDocuments*, "I have to collect the documents I'll have to deliver"). All these predicates are composed by fuzzy operators, and contribute to compute a motivational state for each behavior. The agent knows that it could play a set of behaviors (those enabled by the CANDO conditions), but it has to select among them the behaviors consistent with its present motivations.

3.1 Output generation

Each behavior module receives data in input and provides an output to be addressed to the actuators. In BRIAN, we do not make any hypothesis about the implementation of a behavior module. In general, it can be viewed as a mapping from input variables to output variables:

$$B_i(I_i) : I_i \mapsto A_i \quad I_i \subseteq I = \{i_j\},\ A_i \subseteq A = \{\vec{a}_k\}, \tag{1}$$

where I_i is the set of input variables for the module, and A_i is the set of its actions.

In the application that we will discuss in the rest of the paper, behavior modules are implemented as fuzzy logic controllers (*FLC*): a set of fuzzy rules matches a description of the situation given in terms of fuzzy predicates, and produces actions for the actuators by composing the output proposed by each rule by a T-conorm. Other implementations are possible, for instance as neural network modules, mathematical models, or generic computer programs.

BRIAN first computes the CANDO conditions and selects the behaviors that can be activated, since they have a CANDO value μ_i^c higher than a given threshold τ, which can be defined by the designer or even learnt by experience [16]. Then, BRIAN computes the A_i produced by the selected behaviors associating each of them with the respective μ_i^c value. Finally, the motivation conditions are evaluated, and the result $\mu_i^w$is used to weight the actions a_{ik}, together with the CANDO values. The final action a^f comes from the vectorial composition of the so-obtained values.

$$\vec{a}^f = \sum_{A_i \in A} \left\{ \sum_{\vec{a}_{ik} \in A_i \,\wedge\, \mu_i^c > \tau} \mu_i^c \cdot \mu_i^w \cdot \vec{a}_{ik} \right\} \tag{2}$$

This is the implementation we are adopting in the application we are presenting. However, in BRIAN, we do not make any commitment about the composition of the output from the behavior modules. In particular, alternatively, it would also be possible to select the best action, according to the motivation values, with a formula like:

$$\vec{a}^f = \sum_{\vec{a}_{ik} \in A_i} \vec{a}_{ik} : argmax_i \, (\mu_i^c \cdot \mu_i^w), \tag{3}$$

where $argmax_i$ is the standard function that returns the value of its sub–index (i in this case) for which its argument is maximum. The first approach is followed by the majority of the existing fuzzy behavior management systems [4] [3] [8], since it is analogous to the traditional way of composing output from fuzzy rules; a similar method is adopted also in other non–fuzzy architectures [2]. However, at least in principle, in behavior design, all the possible interactions with other behaviors should be taken into account since the vectorial combination of two actions may produce undesired effects (think, for instance, at the action that may result from the combination of the actions proposed by the *AvoidObstaclesFromLeft* and the *AvoidObstaclesFromRight* behaviors, when facing an obstacle). In principle, this design approach is in contrast with the behavior–independency principle, fundamental in the behavior–based approach to robot control design. The second solution, that is selecting the action proposed by the best fitting behavior, prevents the possibility to pursue multiple goals in parallel.

Both the approaches are implemented in BRIAN and it is possible to select the best one for the application; the behaviors we will show in section 5 are composed by vectorial sum, but we have carefully designed the motivation conditions in order to avoid the activation of two incompatible behaviors at the same time, to avoid the accomplishment of partially fulfilled opposite requirements. This implements a sort of compromise between the two composition methods above mentioned, and can be applied in any domain to have at the same time behaviors whose outputs are composed, and behaviors which are incompatible each other.

4 An application: playing robotic soccer

We have applied our approach both in service and edutainment robotics. Here, we focus on this last, namely, robot soccer playing, since a Robocup [9] match provides a rich and challenging environment where our approach clearly shows its effectiveness.

Robocup is an initiative bringing together hundreds of researchers every year, competing on common testbenches concerning the implementation of effective autonomous robots. There are different competitions, and we describe here the results we have obtained in the F-2000 league of real robots. Here, each robot can have a maximum width and length of about 50 cm. It should be able to play soccer autonomously, by interacting with 3 teammates and 4 competitors in a field sized 4x9 meters. The ground is a green carpet surrounded by white walls, the ball is reddish, all robots should have a "mostly" black body and a cyan or purple marker (a color for each team) 10 cm high, visible from any direction and positioned between 30 and 60 cm above the ground. The ball and the robots can move at more than 1 m/sec. At the first Robocup World Championships (Paris-98 [17], Stockholm-99 [9]) the main challenge was to detect the relevant elements on the field (ball, opponents, goals) and decide reasonable actions in accordance. In the last World Championship [18] some non trivial behaviors have been seen, and the issue of designing appropriate behaviors emerged as really important. Some robots began to show behaviors which seem close to those of human players.

Our players Rullit and Rakataa (respectively represented in figure 3(a) and (b)) have been part of the Italian national soccer team ART (Azzurra Robot Team [19]. We implemented for them ten fuzzy behaviors whose composition enables the robots to play effectively in a match, fighting for the possess of the ball, avoiding walls, other robots, and fouls, kicking in the opponents' goal, and taking a defense behavior when the own goal-keeper has problems. The ten basic behaviors are: *AlignRight*, *AlignLeft*, *GoToGoal*, *Kick*, *AvoidObstacle*, *AvoidWall*, *DefendLeft*, *DefendRight*, *GetOffOwnArea*, *Midfield*.

All our robots, although mechanically different from each other, are equipped by an omnidirectional vision sensor [11], which provides every 100 msec information about the position of the other robots, the ball, and relevant elements of the field. This information is fused in a map with analogous information coming from other robots through the radio ethernet link. The presentation of this complex process is beyond the scope of this paper and has been described elsewhere [14]. The fuzzy predicates used by BRIAN are evaluated from the map.

We can identify subsets of behaviors which are intended to be activated in

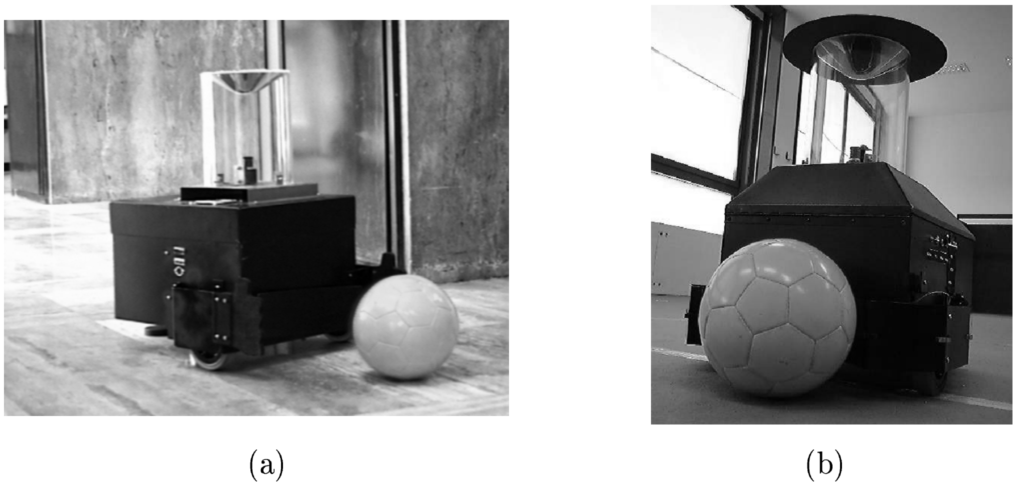


Fig. 3. Our robots Rullit (a) and Rakataa (b).

sequence, and their conditions have been designed to avoid interference. For instance, *AlignToBall* aligns the robot to the direction of the line between the ball and the opponent goal (all our robots have an omnidirectional vision system, so they almost always know where the ball and the goal are); *GoToBall* brings the robot on the ball when it is in the forward direction. The CANDO conditions for *AlignToBall* include the fact that the ball is visible and that the robot is not aligned nor controls it. Notice how these are essential conditions for the behavior. Among the WANT conditions for *GoToBall* we have that the ball should be in the forward direction. The *AlignToBall* behavior tends to make this condition true, and, when this is the case, the context is favorable to the activation of *GoToBall.* Both the behaviors cooperate to bring the agent in a position from where it can take the ball and bring it towards the goal. The *GoToGoal* behavior has among its CANDO conditions predicates corresponding to have the control of the ball and to be aligned to ball and goal: so it can be activated only when both the other mentioned behaviors have achieved their goals.

We have also defined behaviors to take care of the integrity of the robot: these inhibit the others in order to handle critical situations. They are devoted to solve possible problems such as the presence of obstacles or walls in the desired movement directions. We have designed the enabling conditions for these behaviors to implement exclusive activation (as discussed in details in the next section). In particular, all the WANT conditions of the incompatible behaviors contain the control predicate stating that none of the "avoid" behaviors should be active. In this way, if the robot has to avoid something, it does this without any interference from the other behaviors, was action is considered as undesirable when an "avoid" behavior is active.

Interesting behaviors emerge from the interaction of behaviors belonging to the two sets. For instance, we have seen Rakataa dribbling a couple of opponents due to the appropriate switching between *AvoidObstacle*, *Align* and *GoToGoal*. The co-operation of these behaviors made the robot react, when facing an opponent, by throwing the ball aside (obtained by a fast rotation decided by the *Avoid* behavior in order to get around the obstacle) and running to catch it again (composition of *Align* first and *GoToGoal* then).

A second emergent behavior is a sort of defense behavior made by the co-operation between *AvoidWall*, *AvoidObstacle* and *Defense* behaviors. When the ball is close to the wall with another robot that is trying to catch it, our robot stays still, covering its goal area, watching the other robot and waiting that it gets rid of the situation, since *AvoidWall* prevents the robot to get close to the wall. Once the opponent brings the ball away from the wall, Rakataa is ready to close it on the wall again, by applying a *Defend* behavior. Notice that the F-2000 rules state that when the ball is not moving for more than 10 seconds (e.g., because a robot is blocking it against the wall), then it should be removed by the referee and positioned in another position. The mentioned behavior gives also the robot the possibility to stay away from the wall and be ready to get to the new ball position when it is re-positioned.

The third emergent behavior we mention here makes the robot catching the ball near an opponent or a wall. It is obtained by the co-operation between *Align* and *AvoidWall* or *AvoidObstacle*. When the ball is close to a wall (without any other robot in the neighborhoods) or close to another robot, the switching between behaviors makes the robot approaching slowly the ball (since it is close to it) with fast and impulsive rotations due to activation of the *AvoidWall* behavior which tries to avoid the contact with the wall by turning the robot body. When it is close enough to the ball, the result of the interaction between these behaviors is a sort of kick with the robot's hands that throw the ball away from the obstacle.

5 Experimental results

In this section we present in details the results obtained in a particular experiment where the robot executes a standard Robocup challenge used in this case to test the composition of behavioral modules. In this application, each behavior module is implemented by fuzzy rules that infer actions from ground and complex predicates. This implementation is not mandatory, since BRIAN can manage any kinds of behavioral modules (PID control modules, neural networks, etc.) each related to its sets of CANDO and WANT fuzzy conditions, used to select the behaviors to be activated and to compose them.

In figure 4(a) and 4(b) you can see the experimental setting of the test in two different moments and the trajectory executed by the robot. The black

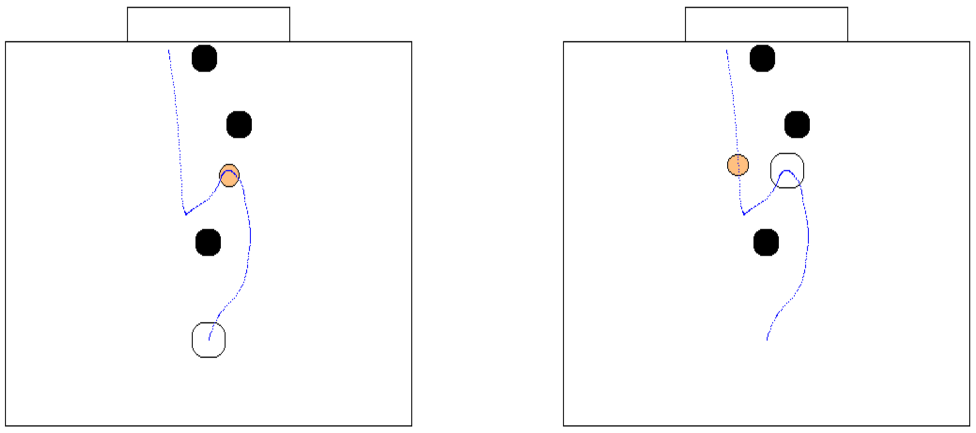

(a) (b)

Fig. 4. The trace of robot trajectory during the test and the position objects at start (a) and after 0.6 (b) seconds.

objects are static obstacles the grey one is the ball to catch and kick in the goal. Track in the plots are taken from real data; we use half of a standard Robocup playground during the challenge, and the trajectory is projected on the field using the odometry of the robot. You can see the initial condition of the test in figure 4(a), the position of our robot (the white block) and the trajectory executed to catch the ball (the gray object in the picture) and kick it in the goal. After 0.5 sec. the robot, incidentally touching the ball, moves it to the position plotted in figure 4(b). So it has to dynamically change its behavior to accomplish this unforeseen situation. The trajectory executed by the robot is obtained by selecting and blending different actions proposed by different behavioral units and reacting to changes in the environment (e.g., the variation of ball position).

In this experiment we use only six simplified basic behaviors to show the effectiveness of our approach. BRIAN selects and combines actions proposed by different modules using fuzzy logic and fuzzy operators. We present their description introducing a preliminary methodology for the definition of CANDO and WANT conditions.

- *AvoidObstacle*: the robot avoids the collision with obstacles when they interfere with the present moving direction; the behavior implemented by fuzzy rules in this module gives results similar to those of an implementation of an *Artificial Potential Field* [20]. It is always active when the robot is moving autonomously, but it make sense only if there are some obstacles to avoid.
- *AlignRight*: if the robot is not aligned with respect to the goal-ball direction it moves to reduce this displacement; this behavior is activated when the robot is in the left part of the field with respect to the goal-ball direction

Behavior	**CANDO**	**WANT**
Avoid Obstacle	`(AND (AutoMode) (ObstaclePresent))`	`(AutoMode)`
Align Right	`(AND (AutoMode) (AND (AND (BallSeeing) (NOT (RigthAligned))) (NOT (AND (BallOwner) (Aligned)))))`	`(AND (AND (AutoMode) (OR (AlonePlayer) (ForwardRole))) (AND (NOT (ObstacleAvoiding)) (NOT (GoalNear))))`
Align Left	`(AND (AutoMode) (AND (AND (BallSeeing) (NOT (LeftAligned))) (NOT (AND (BallOwner) (Aligned)))))`	`(AND (AND (AutoMode) (OR (AlonePlayer) (ForwardRole))) (AND (NOT (ObstacleAvoiding)) (NOT (GoalNear))))`
Go to Goal	`(AND (AutoMode) (AND (BallOwner) (Aligned)))`	`(AND (AutoMode) (AND (NOT (AbstacleAvoiding)) (NOT (GoalNear))))`
Kick in Goal	`(AND (AutoMode) (AND (BallOwner) (Aligned)))`	`(AND (AutoMode) (GoalNear))`
Manual Move	`(NOT (AutoMode))`	`(NOT (AutoMode))`

Table 1
CANDO and WANT conditions for each behavior

(that is the ball is on the right of the robot). The robot "wants" to align when it is moving autonomously (`AutoMode`), it is not avoiding obstacles, and it is the forward role player or the only player in the field. It make sense to *AlignRight* if the robot is not aligned having the ball or does not see the ball.

- *AlignLeft*: is the symmetric of *AlignRight*.
- *GoToGoal*: when the robot owns the ball, it goes in the goal direction, pointing the free portion of it. In order to go towards the goal the robot has to be aligned to the goal owning the ball and in `AutoMode`, it has to do that if it is not avoiding an obstacle or it is not near the goal. In fact, it has to stop when it is near and then kick.
- *KickInGoal*: when the robot is close enough to the goal, executes the action corresponding to activate the mechanical kicker. The robot kicks when it is close to the goal, but only if it owns the ball.
- *ManualMove*: this module implements the manual command for the robot and it is used to control the robot when it is not in autonomous mode. This behavior is active if and only if the robot is not in `AutoMode`. The use of `AutoMode` predicate is to ensure that *ManualMove* and other behaviors are mutually exclusive.

In this particular setting we use simplified CANDO and WANT conditions for each behavior; we present in table 1 such conditions in the lisp-like notation we use in BRIAN.

The predicates appearing in the fuzzy conditions are computed by evaluating fuzzy sets on data provided by the map, and composing them. In figure 5, you may see the definition of some of the fuzzy sets involved in the definition of the above mentioned predicates. In particular, we have reported the frame

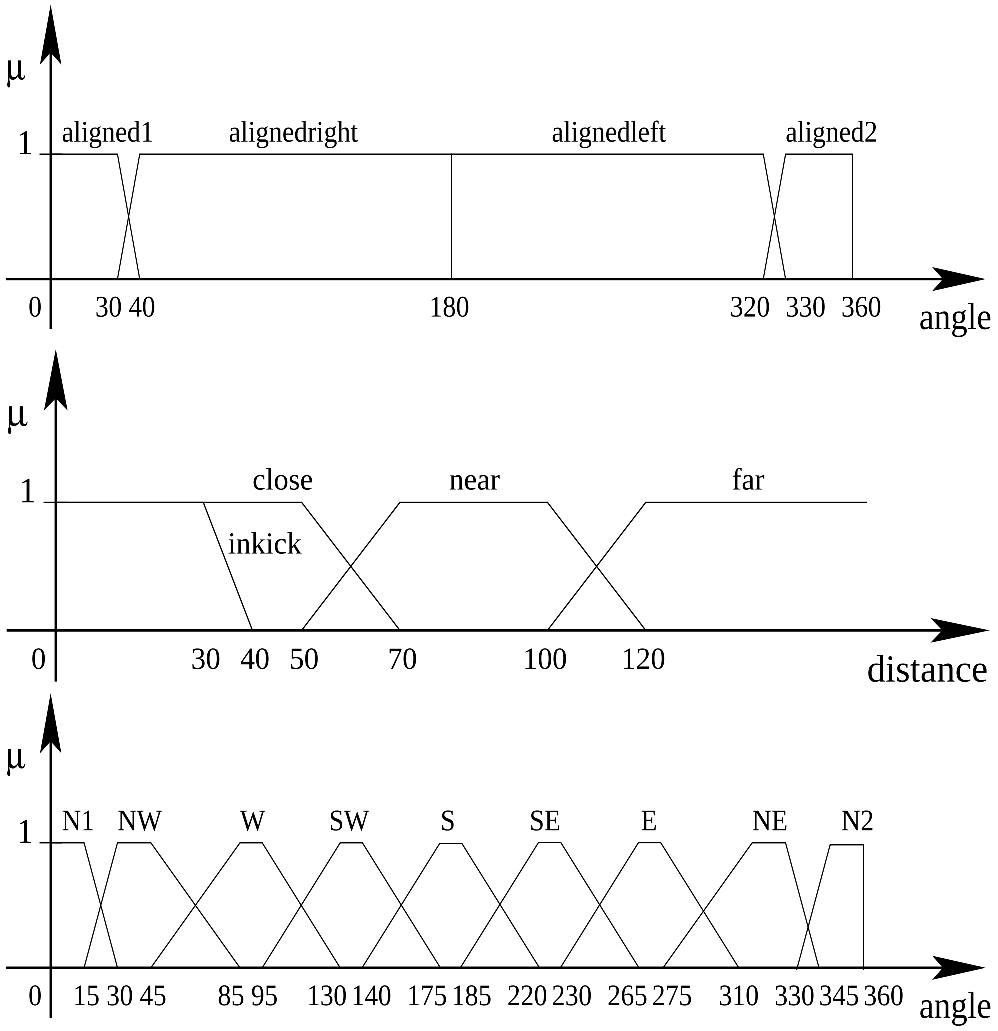
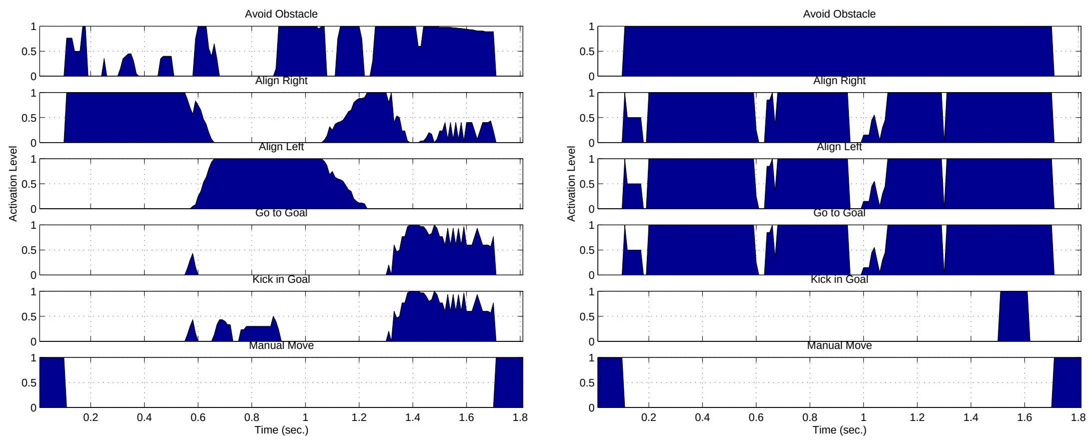


Fig. 5. The definition of some of the fuzzy sets used to compute the predicates involved in CANDO and WANT conditions of the behavior modules developed for Robocup.

of cognition of the alignment with the ball (covering the angle between the heading of the robot and the direction of the ball), the distance to the ball, and the direction of the ball (another frame of cognition covering the same angle mentioned before).

The `BallOwner` predicate is computed as `(AND BallNord BallInKick)`, where `BallNord` comes from `(OR N1 N2)` computed from the third frame of cognition reported in figure 5, and `BallInKick` comes from the value of `InKick` from the distance frame of cognition. The `Align` predicate is computed as `(OR Aligned1 Aligned2)`. Analogous computations bring to the evaluation of all the needed predicates.

As you may notice from the definitions of the WANT conditions of *AlignLeft* and *AlignRight* in table 1, it is possible to include in them also context descrip-



(a) (b)

Fig. 6. The activation level of CANDO (a) and WANT (b) conditions during the test.

tion predicates like the actual role of the teammate (i.e., `(ForwardPlayer)`) or the explicit reference to other behaviors (i.e., `(ObstacleAvoiding)`).

During the experiment we are presenting, we have logged the activation level of CANDO and WANT for each behavior module (figure 6(a) and (b)) and the final behavior activation level coming from the combination of CANDO and WANT (figure 7). From figure 6(b) you may notice that, during this experiment, the WANT conditions for *Align Left*, *AlignRight* and *GoToGoal* have the same activation level, since the robot is playing alone so the predicate `(AlonePlayer)` is always verified and the other conditions are the same for all the behaviors.

We can analyze the global behavior of the robot by looking at figure 7 and considering the traces in figure 4. The robot switches in `AutoMode` after 0.1 seconds than starts the alignment procedure with the *AlignRight* behavior until it reaches the ball and displaces it, at time 0.6, trying to avoid the frontal obstacle. During this part of the test the first obstacle is not considered so relevant due to its distance. Once the robot misses the ball it has to restart the alignment to the ball, this time using the *AlignLeft* behavior. Now, the first robot is perceived as an obstacle for the maneuvering task and this produces a behavior obtained by selection and blending of different modules (*AlignRight*, *AlignLeft* and *AvoidObstacle*) from time 0.9 to time 1.3. At this time, it is aligned to the goal and the *GoToGoal* behavior controls the robot until the *KickInGoal* one activates the mechanical kicker. During this phase the goalkeeper is considered as an obstacle so the active *AvoidObstacle* behavior influences the straight trajectory proposed by the *GoToGoal* making the robot turning slightly left. This results in an optimal strategy to approach the kicking moment. After that, at time 1.7 the robot switches out of `AutoMode`.

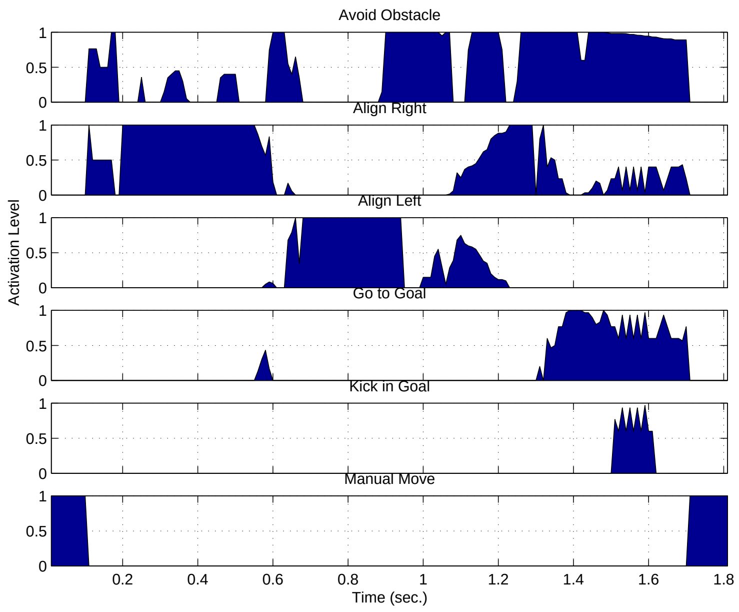


Fig. 7. The logical value of behavioral unit activation during the test

6 Related work

Our architecture follows the Saffiotti's approach [3][8] to the use of fuzzy logic in robotics [4], which tries to face the problem of designing an effective controller for mobile robots by combining goal-specific strategies by resolution of conflicts between multiple objectives. We keep separate the CANDO from the WANT conditions, due to their different semantics, while in [3] they are put together in the *desirability* parameter. Keeping separated those conditions is important for design flexibility and has cognitive plausibility e which makes also the design process closer to the designer way of thinking.

Our behavior architecture is quite different from the Brooks'one [1]: our behavior management system works on fuzzy predicates obtaining the capability of coordinating the concurrent execution of several behaviors. We consider the adoption of CANDO and WANT predicates as an alternative choice to the implementation of the subsumption architecture. It is more general than Brook's proposal and also more effective in strongly dynamic environments. In our implementation the enabling connections among behaviors are context dependent, so relationships among behaviors are not rigidly defined, but we can adapt the emerging global behavior, depending also on external conditions and motivations.

Another reference we have considered is Arkin's schema-based behavior architecture [2]. It is possible to map the basic principles of our and Arkin's approaches into each other. The main difference is the fuzzy model we put at the basis of our architecture, whereas the analogous features are represented by Arkin by different tools such as a *gain* value to weight each behavior contribution, and their representation in terms of potential fields. However, the gain has a different meaning, stating the a priori relevance of a behavior with respect to another, while we blend (or select) behaviors according to their condition values and context in any time instant.

Another difference between our architecture and both Brooks' and Arkin's is the presence in ours of a world modeler that interfaces the external world with the behavior module. In the mentioned architectures, world modelling is embedded in each behavior definition. We have implemented such a module to achieve efficiency and to provide a unified interface from the environment to the behaviors.

7 Conclusion

We have presented in this paper a general behavioral architecture for autonomous robots based on fuzzy models. We have focused on the behavior management module, whose most relevant feature is the use of fuzzy enabling and motivation conditions. It has been designed to make it possible different types of interaction among behavior modules and to face virtually all application environments. Fuzzy models give robustness to the system with respect to inaccuracy in data acquisition and uncertainty, and make quite easy the definition of behaviors and their interaction.

Learning may support the development and adaptation of robotic agents. In particular, we have shown elsewhere [16],[21] how it could be possible to adapt behaviors in a short time to new environments by tuning their context predicates. The architecture here presented is essential in this activity, since it makes distinct the different aspects (CANDO, WANT, and behavior code) which can be separately designed, learned or adapted. We are presently working at an extensive use of learning mechanisms working on these aspects to adapt in real time the robot behaviors to different environmental conditions.

References

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

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

[3] A. Saffiotti, K. Konolige, E. H. Ruspini, A multivalued-logic approach to integrating planning and control, Artificial Intelligence Journal 76 (1-2) (1995) 481–526.

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

[5] F. J. Radermacher, Cognition in systems, Cybernetics and Systems 27 (1996) 1–41.

[6] G. J. Klir, B. Yuan, U. S. Clair, Fuzzy set theory: foundations and applicatons, Prentice-Hall, Upper Saddle River, NY, 1997.

[7] L. A. Zadeh, Making computer think like people, IEEE Spectrum (1984) 26–32.

[8] K. Konolige, K. Myers, E. Ruspini, A. Saffiotti, The saphira architecture: A design for autonomy, Journal of Experimental and Theoretical Artificial Intelligence 9 (1) (1997) 215–235.

[9] M. Asada, H. Kitano, I. Noda, M. Veloso, Robocup: today and tomorrow – what we have learned, Artificial Intelligence Journal 110 (1999) 193–214.

[10] Y. Yagi, S. Kawato, S. Tsuji, Real-time omnidirectional image sensor (copis) for vision-guided navigation, IEEE Transactions on Robotics and Automation 10 (1) (1994) 11–22.

[11] A. Bonarini, P. Aliverti, M. Lucioni, An omnidirectional vision sensor for fast tracking for mobile robots, IEEE Trans. on Instr. and Meas. 49 (3) (2000) 509–512.

[12] P. Lima, A. Bonarini, C. Machado, F. Marchese, C. Marques, F. Ribeiro, D. Sorrenti, Omni-directional catadioptric vision for soccer robots, Journal of Robotics and Autonomous Systems .

[13] S. Coradeschi, A. Saffiotti, Anchoring symbols to vision data by fuzzy logic, Lecture Notes in Computer Science 1638 (1999) 104–112.

[14] A. Bonarini, M. Matteucci, M. Restelli, Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem?, in: Proceedings of the AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems, AAAI Press, Menlo Park, CA, 2001, p. In press.

[15] G. L. Drescher, Made-up minds: A constructivist approach to artificial intelligence, Ph.D. thesis, MIT EECS Department, (Reprinted by MIT Press.) (Sep. 1989).

[16] A. Bonarini, M. Matteucci, Learning context motivation in coordinated behaviors, in: Proceedings of IAS-6, IOS Press, Amsterdam, NL, 2000, pp. 519–526.

[17] M. Asada (Ed.), RoboCup-98: Robot Soccer World Cup II, Springer Verlag, Berlin, D, 1998.

[18] P. S. G. Kraetzschmar, T. Balch (Eds.), RoboCup-2000: Robot Soccer WorldCup IV, Springer Verlag, Berlin, D, 2001.

[19] D. Nardi, G. Adorni, A. Bonarini, A. Chella, G. Clemente, E. Pagello, M. Piaggio, Art – azzurra robot team, in: M. A. M. Veloso, E. Pagello (Ed.), RoboCup 98–Robot Soccer World Cup III, Springer Verlag, Berlin, D, 2000, pp. 695–698.

[20] C. W. Warren, Global path planning using artificial potential field, in: Proceedings of IEEE International Conference on Robotics and Automation, Vol. 1, IEEE Computer Press, Piscataway, NJ, 1989, pp. 316–321.

[21] A. Bonarini, Evolutionary computing and fuzzy rules for knowledge acquisition in agent-based systems, Proceedings of IEEE In press.