EVOLUTIONARY LEARNING OF FUZZY RULES IN A MODIFIED CLASSIFIER SYSTEM FOR MOBILE AGENTS CONTROL

Eric Vallejo Rodríguez
GRSI, Universidad del Norte, Km.5 Vía a Puerto Colombia, Barranquilla, Colombia

Ginés Benet Gilabert
Dpto. de Informática de Sistemas y Computadores, DISCA, Universidad Politécnica de Valencia, Valencia, España

Keywords: Fuzzy control, Evolutionary Computing, Machine Learning, Artificial Intelligence, Classifier Systems.

Abstract: In this work we present the creation of a platform, along with an algorithm to evolve the learning of FLCs, especially aiming to the development of fuzzy controllers for mobile robot navigation. The structure has been proven on a Kephera robot. The conceptual aspects that sustain the work include topics such as Artificial Intelligence (AI), control advanced techniques, sensorial systems and mechatronics. Topics related with the control and automatic navigation of robotic systems especially with learning are approached, based on the Fuzzy Logic theory and evolutionary computing. We can say that our structure corresponds basically to a *Classifier System*, with appropriate modifications for the objective of generating controllers for mobile robot trajectories. The more stress is made on genetic profile than in the characteristics of the individuals and on the other, the strategy of distribution of the reinforcement is emphasized, fundamental aspects on which the work seeks to contribute.

1 INTRODUCTION

The autonomous robot navigation in non-structured environments is one of the most important technological challenges in the field of mobile robotics. For this reason, the development of techniques of control and navigation concentrates the biggest investigators' efforts. The real world is generally non-structured and dynamic; therefore the robot should acquire dexterities along with security and robustness to face non-adjusted environments.

Many strategies and paradigms for the solution of the control problem and mobile robots' navigation in such environments have been proposed. One of the alternatives for the development of such controllers is that of learning. The motivation that impels these works is the long-term vision of achieving robots easy to use in the real world. To reach this, robotic systems should be intelligent, flexible, and the most important thing, easy to program in an interactive way. It is also very possible that robots acquire "*their learning*" in environments different from those in which it will finally act, therefore that learning will be made, in similar environments but not necessarily identical. The above-mentioned implies big challenges in different directions, one of them being that related with learning, its validation and the simplicity in its implementation.

In this work we have aimed to improve and to apply an algorithm on a software platform for the generation of controllers with fuzzy inference for learning, developed especially for this purpose. The platform facilitates to develop control systems with blocks of easy interconnection and depuration, and it also allows the validation operating directly on a simulated or real robot. For that, we have taken a very well known mini-robot in the environment of the groups of I+D in mobile robotics: the *Khepera*, developed by the K-Team. On the other hand, we have had software tools for design and simulation provided by robot's makers as well as by other sources, besides MATLAB and some of their toolboxes.

This article is divided in four parts: in the first one the necessary aspects to establish an appropriate

Vallejo Rodríguez E. and Benet Gilabert G. (2005).
EVOLUTIONARY LEARNING OF FUZZY RULES IN A MODIFIED CLASSIFIER SYSTEM FOR MOBILE AGENTS CONTROL.
In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics - Robotics and Automation*, pages 62-69
DOI: 10.5220/0001189000620069

conceptualization of the work are presented. In the second, some of the used tools are described. The third part shows the experimental realization and results are exposed and in the last one the conclusions and proposals for later work are offered.

2 INTELLIGENT CONTROL

2.1 Artificial intelligence

The concept of Artificial Intelligence (AI) can have different connotations, depending on the source taken as reference and of its point of view. But if one thing is clear: AI has as one of its objectives the study of the intelligent behaviour of machines (Nilsson N., 2001). Another of its pursued purposes is the development of machines able to execute in a similar -or better- way tasks developed by biological organisms and to understand that behaviour. In short, we can say that AI looks for scientific, philosophical and engineering objectives.

It is generally considered that a machine is intelligent if it is able to successfully carry out similar processes such as those made by biological entities, based on in imprecise, qualitative and not very reliable reasoning that also modifies and adjusts continually with the acquisition of new information.

2.2 Intelligent controllers

The term intelligent control refers to the focus of control design as well as to the installation of techniques of AI that emulate certain characteristics of intelligent biological systems (De Andrés, 2002). The intelligent control has allowed recapturing old control problems to give them solution and to confront other new ones, thanks to the growing computational capacities. Maybe the paradigm of the intelligent control is focused on approaches of biological inspiration, without making clear that the resulting controllers perhaps drastically differ from them and that they often seem simply adaptive controllers (Athans, 1998). The Expert Systems in systems of direct control, for example, have as predecessor techniques of conventional control with controllers based on automata, Petri nets and other discrete event systems (Passino, 1996).

2.3 Control of mobile robots

For some decades some alternatives have been considered for the control and guidance of mobile robots in non-structured partially structured environments, the natural environment for the yearned applications of these machines. All of them could be summarized in three big groups or paradigms: hierarchical, reactive and hybrid paradigms (Murphy, 2000), as we see in figure 1.

In hierarchical systems planning includes a modelling, and they can render an optimal solution if the problem and the environment are very defined and the environment doesn't change. In reactive control system, sensors are directly related to control actions; planning does hardly exist. These machines constitute the basic vehicles of Braitenberg (Braitenberg, 1984) that although they guarantee a quick answer, the same does not happen in optimization. They are the bases of architectures based on behaviours. Hybrid structures possess a reactive base at a low level and a planning block at a superior one. The planning block generally contains a group of simple tasks (behaviours (Brooks, 1985)) that are activated in a dynamic way to endow the system with intelligent behaviour; it is a level of deliberative control.

2.3.1 Fuzzy control in robotics

Formal reasoning ends in definitive conclusions; common reasoning ends in provisional conclusions (Gulley, 1995). It is difficult to achieve the systematic representation of ambiguity and uncertainty, characteristics of common knowledge. Fuzzy Logic allows the representation, in some way,

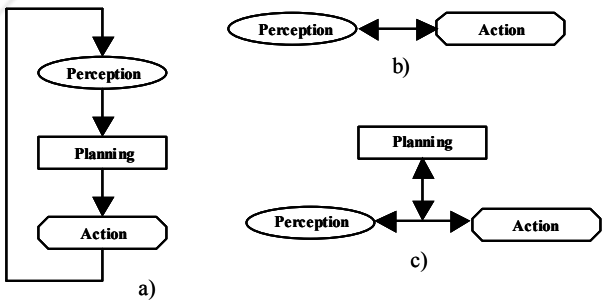

Figure 1: Structures of robots' control: to) hierarchical, b) reactive, c) hybrid

of uncertainty and ambiguity. This is why it has been used in artificial intelligent systems as tool to transfer them common knowledge and experience.

It would be oppressive to mention the numerous works that demonstrate that the techniques of fuzzy control turn out to be very effective for handling reactive mobile robots, because translating values of sensors in control actions is immediate with rules of inference in the way IF - THEN, characteristic of Logic. Let us take as an example the inference

***IF** (x is A) **AND** (y is B) **THEN** (z is C)*

that it could be translated in an elementary behaviour such as "*If there is an obstacle in front and there is free space to the right then, turn to the right*". On the other hand, if the environment in which a mobile robot performs is non-structured, the control system should be able to deal with the uncertainty and the ambiguity, space where the fuzzy logic, find their largest action field.

2.3.2 Learning

Learning eliminates the programming of specific algorithms, building intelligence through experience in a very similar way to that of biological methods. As for the methodology of machine learning strictly speaking, it can be summarized that the base of knowledge can be modified basically in two ways: with structural modifications (algorithmic reconfiguration) or with parametric modifications (adjusting the parameters of the system). When the systems are very complex or a model of the system cannot directly be obtained, the modification of the base of knowledge is far from the classic techniques of control. In consequence, countless proposals in the topic of learning have been presented.

Some systems based on knowledge, as the Fuzzy Systems and the Expert Systems, find application in automating characteristics of perception, knowledge and decision taking characteristic of human operators. Artificial Neural Nets (ANN) slightly emulate Natural Neural Nets (NNN). They have been used, for example, to learn the form of controlling a system "*observing*" the actions taken by an operator. Genetic Algorithms (GA) are used in computer aided design "*to evolve*" controllers under the survival principle for those which better adjust to a certain objective.

3 ROBOTS SIMULATION

Thanks to simulators, it is possible to design and prove control strategies in a quick way, observing and correcting actions or inadequate behaviours without putting the robot or the environment at risk. However, it is very desirable that what is developed in a simulator can be moved immediately way and without any effort to the real environment.

A great availability of programs exists for the simulation of robots; some are commercial ones and others of free distribution. Here we will refer especially to two programs that we have used together: the KMatLab, created by the K-Team (K-Team, 1999) and KIKS, developed by Theodor Nilsson (Nilsson T., 2001).

3.1 KMatLab

KMatLab was created to operate the Khepera from MatLab through a series port of the computer. It is a group of libraries *.dll* for Windows designed to configure double communication between the robot Khepera and MatLab. It also incorporates a set of libraries under MatLab to execute the instructions of low level belonging to the Khepera. The set of instructions of KMatLab possesses mnemonics for the instructions that make them easier to learn and follow in the program listing. For example, an instruction of KMatLab has the form

KSetSpeed(ref, left, right)

where *ref* is the variable in MatLab that represents the address from the port series to which the robot and other parameters of the communication are connected. Additionally, left and right represent the values of speed with which the respective controllers of the robot's motors should be loaded. In fact, the instruction of the example is completely equivalent to instruction *D* of the Khepera (K-Team, 1999).

3.2 KIKS

KIKS (***K**iks **I**s a **K**hepera **S**imulator*) it is a program that allows interacting with a simulated Khepera robot. KIKS is in fact a KMatLab simulator, although it also allows communication with a real robot. For this it uses instructions that call to those equivalent ones in KMatLab with the possibility of defining it a virtual port for the simulation. As example of the instructions of KIKS we take

ref = kiks_Kopen([port, baud, 1])

that indicates the system the port that will be enabled for the robot, the communication speed and the time out for the communication. This instruction is exactly equivalent to

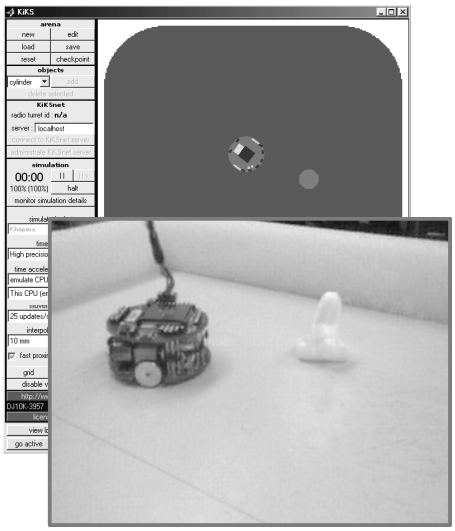

Figure 2: Simulated and real environments for Khepera robot

ref = Kopen([port, baud, 1])

of KMatLab. Only very few instructions for the Khepera are not supported by the KIKS.

4 EXPERIMENTAL WORK

In our work we have approached topics related with the control of robotic systems navigation with learning, based on the theory of Fuzzy Logic and Evolutionary Computing. All the exercises are executed on the Khepera robot.

4.1 Platform for design of controllers

In a published work we showed the viability of using support tools to software design (CASD), using an experimentation platform module developed on MatLab, Simulink and some Toolboxes. The idea was to have a platform for I+D in robots' control and sensorial whose designs and final verification could be given in very brief time.

On it were designed and tested behaviour based controllers developed with Simulink and others of fuzzy inference (FIS) integrated in Simulink.

Based on this structure, we can generate fuzzy logic controllers (FLCs) and transfer them to the robot by means of Simulink or directly with MatLab through a port of the PC. If the port is defined as virtual, the controller's actions will go to the simulated environment (KIKS). If the actions are directed through a real port, one will have control on the real robot. When one works from Simulink, the real environment is managed as a simulated environment. In figure 2 the graphs of the simulated (KIKS) and real environments are shown.

4.2 Platform for controller learning

In our experimental work we have decided to look for solutions to the problems of controllers' fuzzy generation for learning using technical of evolutionary computing, creating a method and some tools of "*training*" of mobile robots that allow to validate the acquisition of knowledge on the part of the machine. Concretely we could say that we adopt the proposal of Bonarini (Bonarini, 1996) to which we seek to contribute with some advances. With this general position we saw the necessity to modify our first platform, taking it to what we schematize in figure 3.

Since the MatLab Fuzzy Logic Toolbox doesn't provide some necessary files for the process of adopted learning, we opted to add several functions (*files .m*) that would extract them. This way we can have such parameters as the contribution of each rule to the controller created in each step of the learning, as well as the generated rules and other necessary to the process. With these files we can access, through KMatLab, to any of the two environments: simulated or real. For our applications we appeal to the version 6.1R12 of MatLab and the Fuzzy Logic Toolbox for this version, although there were not restrictions to use previous or later versions. In hardware it was used a desk PC with an AMD XP2000+ processor and 256 MB of memory.

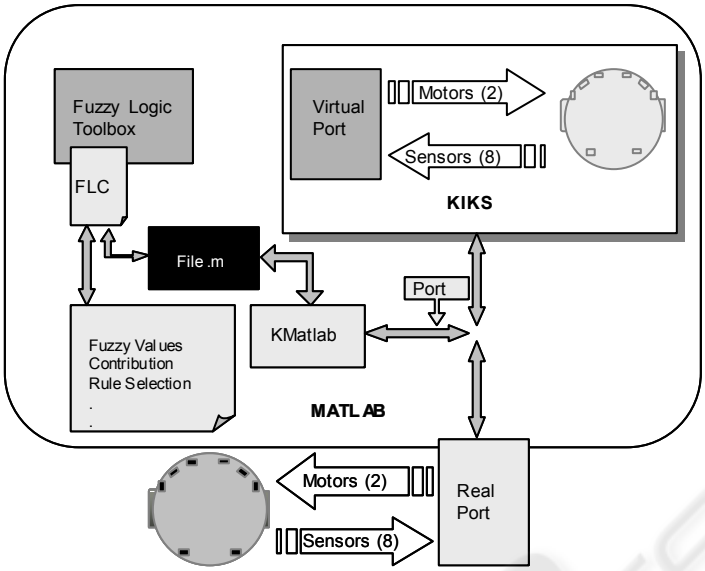

Figure 3: Experimental platform for the generation of controllers for learning

4.3 Evolutionary learning of fuzzy rules (ELF)

The implemented system is based on the ELF algorithm developed by Bonarini (Bonarini, 1994, 96), a Q-Learning algorithm adapted to learn fuzzy controllers (FLCs) in a mobile robot. For doing that we developed the MatLab platform previously described and we looked for more efficient way for such an algorithm incorporating other new concepts. Some significant differences are focused especially on the use of genetic algorithms, performance evaluation functions (PEF) and population's updating, which improves the algorithm outcome facilitating its use in dynamic environments.

In this strategy of evolutionary learning, the mobile robot dynamically learns a set of fuzzy rules. Specifically, the system learns the combination of antecedents and the consequence of the rules to satisfy the requirements specified by the user by means of an evaluation function. Our experimentation is made basically through the mutation operator, since for the particular application it is more than enough, but the developed platform does not limit the use of other genetic operators. The controller is composed by a group of fuzzy rules whose number is adjusted dynamically by the system, maintaining the best ones. The membership functions of the fuzzy sets that represent the variables of the system are fixed. The agent can begin with a set of random rules or previously learned, or without any rule. For different specified tasks, different FLCs are learned. In synthesis, our learning method is of the type of structural modification and it belongs basically to a *Classifier System*, with appropriate modifications for the objective of generating controllers for mobile robots.

4.3.1 The ELF algorithm

This algorithm starts with having a population of rules and it has the following relevant characteristics:

- The total population of rules is divided into sub-populations whose members share the same antecedents (labels).
- In each sub-population we have rules with the same antecedents and different consequents, competing with each other to propose the best consequent for the situation described by the antecedent.
- All sub-populations cooperate to produce the control action.

In a simple way we can synthesize what has been presented with graph 4. In this graph we have divided the population of rules in groups (sub-populations) *a, b, ..., n* that gather the rules possessing the same antecedent but they have different consequent. From each group a rule is taken randomly to generate a FLC that then is proven in the system. Later the contribution of each rule to the actions is valued to grant it the corresponding reinforcement.

In ELF the learning cycle is compound by several trials that in turn are composed of episodes.

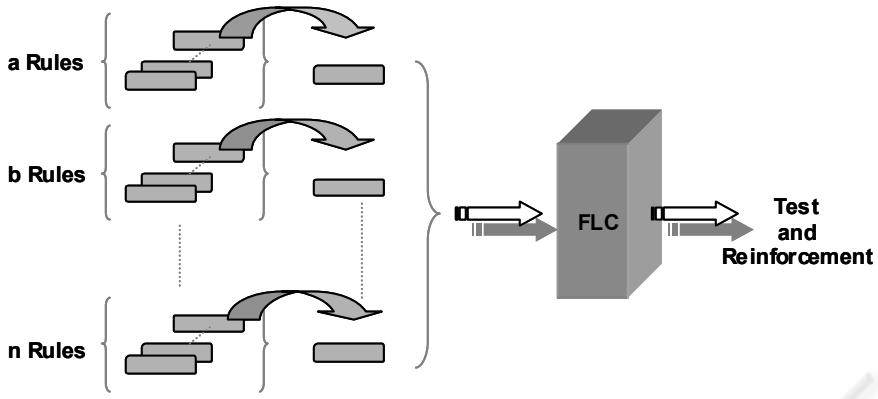

Figure 4: Graphic representation of the ELF algorithm

Each episode is conformed by control cycles that in fact are detection-action cycles, characteristic of a purely reactive paradigm.

The consolidation (strength) of a rule is given by:

$$s_r = s_r(t-1) + (reinforcement(t) - s_r(t-1)) * \frac{currc_r}{pastc_r}$$

where s_r represents the strength of a rule in the present cycle, *currcr* is a value between [0 ..1] that represents the contribution of the rule to the actions given during the current episode and *pastcr* it is a measure of the contribution of the activated rule in previous episodes.

On the other hand

$$currc_{\hat{r}} = \frac{\sum_{s \in S(E)} \mu_s(\hat{r})}{\sum_{\substack{s \in S(E) \\ rinR}} \mu_s(r)}$$

Where $\hat{r}$ is the rule under trial, *s* is a state belonging to the group of states *S* visited during the episode *E*, $\mu(\hat{r})$ is the degree of concordance of rule *r* to state *s* and *R* is the set of activated rules.

In order to value the contribution of the rules activated in previous episodes, a reinforcement given by this equation is applied

$$s_r = s_r(t-1) + (reinforcement(t) - s_r(t-1)) * \frac{currc_r}{pastc_r} * decay$$

establishing that there is a correlation between an episode and the past ones. Then

$$decay = correlation$$

Since the number of rules could grow exponentially causing an "*explosion of rules*", a heuristic equation is applied, which maintains an optimal number of them.

$$o\_c = \max\left(1, \left[\frac{Max\_reinf - (Max\_reinf * 0.1) - Max\_vote\_sp}{Max\_reinf * 0.1}\right]\right)$$

Max_reinf is the maximum reinforcement value and *Max_vote_sp* is the maximum value so far obtained by the sub-population's rules.

4.3.2 Proposal

In our work with ELF We are focused on four aspects:

- More generalization capacities.
- Better adaptation to non-structured environments.
- Smaller limitations with dynamic elements.
- Improving of local minima handling.

To reach these objectives, our theoretical work has taken us to consider three possible elements to experience in the algorithm. The first of them is the possibility to place a penalization value in the learning process that could have this form in the case of the mission of obstacles avoidance

$$pain = f\left(\sum_{i=0}^{5} s_i\right)$$

where S_i represents the reading of sensors. The readings of the two front sensors are fused, taking its resultant as that of a unique sensor.

In consideration to the good control actions, we can add the concept of pleasure that would have the form

$$pleasure = \gamma * f(m_l, m_r) + \beta$$

representing m_l and m_r the actions of the left and right motors respectively. γ and β are weighting values.

A point to consider is to speed up the learning consists of revising equation for S_r, since the computational work to for the task. The reinforcement equation for objective searching and obstacle avoidance could have the form:

$$r(t) = g(dist\_inic - dist\_act) * f(dist\_obs)$$

where *dist_act* y *dist_inic* are, respectively, the current and initial distances with respect to the objective and *f* is a function related with the distance to an obstacle.

Once the exit of the system is satisfactory in the learning process, the training stops and the resulting fuzzy controller can be proven in the same environment or in another dynamically changing one. The fuzzy rules are represented as chromosomes and their parts as genes with the purpose of using genetic algorithms for learning. The values of each gene are not strictly 0 or 1, as in genetic algorithms, but rather they vary with the value of membership of each variable. This way, each gene really represents a variable. With our development platform created with MatLab, vectors that compose the matrix that constitutes the controller give the representation of chromosomes. These vectors include the chromosomes already mentioned, and other necessary information in the learning.

4.4 Tests and results

Next we present some results obtained with controllers for obstacle avoidance (including navigation in narrow corridors, a recurrent problem in mobile robotics) and contour following. The graphics show that the obtained controllers operate satisfactorily in the selected worlds.

4.4.1 Obstacle avoidance

One of the most required behaviours in autonomous mobile robotics is navigation with obstacle avoidance. Controllers that value positively the displacement at great speed on straight line avoiding crashes with objects and walls of the environment have been achieved. In figure 5a) we show the robot's behaviour during the learning and its later operation with the learned controller.

The reinforcement equation developed for this task is

$$r = \left(1 - \left(\frac{1}{80} * \left(|vel_l - vel_r| - (vel_l + vel_r) + |vel_l| + |vel_r|\right)\right)\right) * \frac{1}{e^{\sigma}}\left(e^{distn*\sigma} - 1\right)$$

4.4.2 Following of contours

Another very useful task in mobile robotics is the capacity to surround objects. In figure 5b) we present two executions. In the first one we have not placed the valuation to the trajectory on straight line during the training. In second case this behaviour has been rewarded. The used reinforcement equation is

$$r = \left(1 - \left(\frac{1}{80} * \left(|vel_l - vel_r| - (vel_l + vel_r) + |vel_l| + |vel_r|\right)\right)\right) * \left(1 - e^{distn*\lambda}\right)$$

5 CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

- It was obtained with MatLab a modular platform where exchanging of models or environments is immediate and simple.
- The architecture of the control system is stratified and modular, so each module has a low complexity and can be learned in short time.
- The algorithm has been proven in real and simulated environments giving similar results, in accordance with what expected.
- The form of consolidating the rules was studied and we opted for solutions with easy evaluation simple equations.
- With these strategies, and the reduction of the search space, it improves the robustness to data with noise, helped by the representation of the control modules in terms of fuzzy rules.
- The results obtained in our experimental work demonstrate to be successful with sessions inferior to 500 cycles for simple tasks as those shown, observing adequate behaviours in 4 or 5-minute sessions in trainings with the real robot.

5.2 Future work

We seek to take our experiences with the Khepera to other robots, such as YAIR, mobile experimentation robot developed by the Group of Sistemas de Tiempo Real of Computer Science's of Systems and Computers (DISCA) Department of the Universidad Politécnica de Valencia (Spain), and to TELÉMACO, experimental robot of the Grupo de Investigación en Robótica y Sistemas Inteligentes of the Universidad del Norte (Colombia).

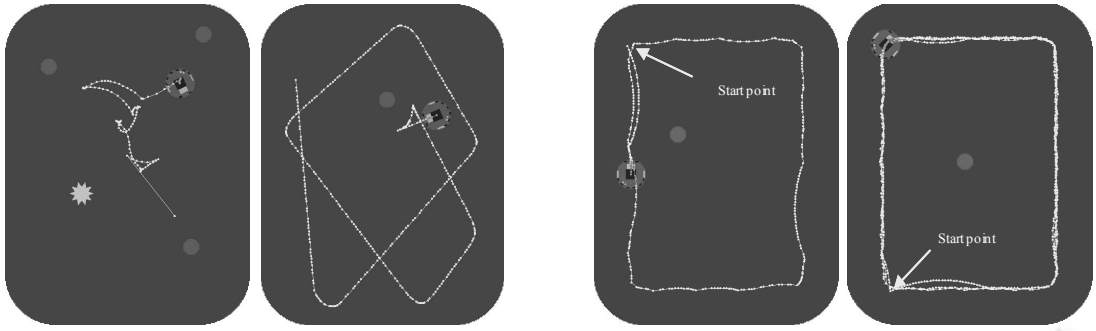

Figure 5: Learning and the controller's verification

With the support of other tools, we will be able to obtain controllers for particular platforms, coded in high-level languages starting from our simulated models. It is possible to carry out these structures in general or of lower level in application languages and to insert them, in even smaller robots that possess very few resources.

We are working to add more complex behaviors and to develop controllers with superior layers.

REFERENCES

Athans, M., 1998 "*Why I Despise Fuzzy Feedback Control*" IEEE CDC/98 Debates, Tampa, Florida.

Bonarini, A., 1994 "*Evolutionary learning of general fuzzy rules with biased evaluation functions: competition and cooperation*". Proceedings. of the IEEE World congress on Computational Intelligence (WCCI) - Evolutionary Computation, IEEE Computer Press, Piscataway, NJ, 51-56.

Bonarini, A., 1996 "*Learning behaviors implemented as Fuzzy Logic Controllers for Autonomous Agents*". Proceedings of the WEC2, University of Nagoya, Nagoya, J, 21-24.

Braitenberg, V., 1984, Fourth printing (1994) "*Vehicles*", MIT press, Bradford Books, England.

Brooks, R., 1985 "*A Robust Layered Control System for a Mobile Robot*", A. I. Memo 864, MIT, Artificial Intelligence Laboratory.

De Andrés, T., 2002 "*Homo cybersapiens: La inteligencia artificial y la humana*". EUNSA, España, ISBN: 84-313-1982-8.

Gulley, N., et. al., 1995 "*Fuzzy Logic Toolbox*". MathWorks.

Murphy, R., 2000 "*Introduction to AI Robotics*", MIT press, Bradford Books, England, ISBN: 0-262-13383-0.

Nilsson, N., 2001 "*Inteligencia Artificial; una nueva síntesis*", Ed. McGraw-Hill, España, ISBN: 84-481-2824-9.

Nilsson, T., 2001 "*KIKS User Guide*". www.tstorm.se/projects/kiks/.

Passino, K., Özgüner Ü., 1996 "*Intelligent Control: From Theory to Application*", Guest Editors' Introduction to the IEEE Expert Special Track on Intelligent Control, Volume 11, No. 2, pp. 28-30.

K-Team, 1999 "*Khepera User Manual*", www.k-team.com.