A Reactive Approach to Classifier Systems

José M. Molina, Carlos Sevilla, Pedro Isasi, Araceli Sanchis
Grupo de Vida Artificial. Departamento de Informática.
Universidad Carlos III de Madrid, Spain.
Avda. Butarque 15, 28911 Leganés. Madrid.

ABSTRACT

The navigation problem involves how to reach a goal avoiding obstacles in dynamic environments. This problem can be faced considering reactions and/or sequences of actions. Classifier Systems (CS) have proven their ability of continous learning, however they have some problems in reactive systems. A modified CS is proposed to overcome these problems. Two special mechanisms are included in the developed CS to allow the learning of both reactions and sequences of actions. This learning process involves two main tasks: first, discriminating between rules and second, the discovery of new rules to obtain a successful operation in dynamic environments. Different experiments have been carried out using a mini-robot Khepera to find a generalized solution. The results show the ability of the system for continuous learning and adaptation to new situations.

1. INTRODUCTION

A Classifier System (CS) [4] is well suited to learn multiple different concepts incrementally under payoff. These systems have been widely implemented and tested for a large number of theoretical problems, [15, 16], but there are not many cases in which they are included in real systems [2, 14, 15]. In the most recent bibliography, especially in [14], the CS's appear as systems of doubtful efficiency learning. They were employed with great frequency from the moment of their description by Holland [4, 5], but today they have smaller summit because of the problems and difficulties that present. When they are intended to apply Classifier Systems to the resolution of certain importance problems, a series of difficulties appear, that they could, even, make to think about the convenience of employing any other system. One of the principal problems located in CS's are related to their application to dynamical environments.

In spite of the article of Brooker [1], that describes these systems as prepared to operate in changing environments (concretely, he has developed a controller system for a predator that moves and hunts in a world), the reality is that in the bibliography are not collected good results for these cases. This bad results are due to the fact that, if the decision time is let to increase to the classifier, something which is necessary to provide a elaborated solution to the problem, while the individual (predator in [1] or "animat" in [15] and [16]) continues inside the world and being moved in it, when the system provides the solution, this no longer results valid, in most of the cases. Furthermore, the evaluation of that decision is not valid, since the appropriateness of the output is not known, due to the temporary lag between input and output.

The problem of the capacity of the system by producing a quick response should not be approached only from techniques that attempt to increase the speed of the process, but they can be approached from a different perspective: the injection of environmental data and obtainment of intermediate decisions in the course of global decision [11, 12]. The rule chaining in the traditional CS makes the system blind to the environment because it can not manage new sensorial inputs during the decision process. In a dynamic environment, system ought to read sensors in each decision step (reaction), that is the main feature of reactive systems. For instance, in a navigation problem in a dynamic environment (where the obstacles are moving around) a robot ought not to be blind any time, so each movement has to be the result of applying the decision process over the last sensorial input.

In this work, a new CS is proposed, Reactive CS (RCS), modifying the general process in order to allow reactions without loosing the possibility of rule chaining. The new process integrates the environmental input with the internal state of the previous input. Then, from an input, the RCS gives directly an action and, at the same time, modified the internal state. When the next input arrives, the message is fused with the previous internal state to allow a new reaction or an action that chains with the previous action.

This new RCS will be used to learn a fundamental requirement for autonomous mobile robots: navigation. This task gets the robot from place to place with safety and no damages. In the proposed learning process, the only previous information is about the number of inputs (robot sensors), the range of sensors, the number of outputs (number of robot motors) and its description. The robot controller starts without information about the right associations between sensor inputs and motor velocities. And from this situation the robot is able to learn through experience to reach the highest adaptability grade to the sensors information. The results obtained proved the capability of generating not only new better rules but the mechanisms for chaining new and existing rules.

Another important aspect verified in this work is the possibility of continuously learning and adaptation to new situations that allow to solve the problem even if there are mobile objects, more than one goal, and dynamical goals that could appear and disappear or move when the robot is navigating.

In this paper we present the results of a research aimed at learning reactive behaviors in an autonomous robot. In section 2, we outline the general theory of classifier and reactive systems. Section 3 is related to RCS and the goals of the work.

0-7803-4778-1/98 $10.00 © 1998 IEEE

Authorized licensed use limited to: Univ Carlos III. Downloaded on March 26, 2009 at 11:41 from IEEE Xplore. Restrictions apply.

In Section 4 we present the particularities of the developed RCS for navigation problem. Some experimental results are showed in Section 5. The last section contains some concluding remarks.

2. CLASSIFIER SYSTEMS AND REACTIVITY

Classifiers Systems, CS, are a specialized form of production systems that have been designed to be specifically amenable to the use of genetic algorithms [3]. These systems were developed by Holland and Reitman [5], and later refined and modeled by Holland [6]. CS are machine learning systems that learn syntactically simple string rules (called *classifiers*) to guide their performance in an arbitrary environment [3]. A schematic representation of a Classifier System is showed in figure 1. In these systems, it can be distinguished three activity levels:

(1) **Performance**, rule and message system, or some times called Classifier System, that interacts with the environment, gathering information through the input interface and producing the output through the output interface; it also receives the payoff. Structurally, the performance level consists of: (A) a finite population of fixed length condition/action rules, (B) a message list, (C) an input interface consisting of a set of environmental feature detectors and (D) an output interface for acting in the environment, that are also shown in figure 1.
(2) **Credit Assignment**, that rewards rules on the basis of their observed utility to the systems goal.
(3) **Discovery**, that employs a genetic algorithm as a discovery operator that automatically generates new rules.

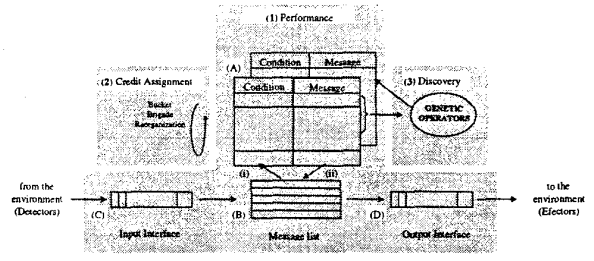

Figure 1: Representation of a Classifier System. (i) All messages are tested with all classifiers. (ii) Winning classifiers post their messages to the message list.

In the performance level, rules are composed of two parts: condition and message. Rules are codified as strings: each condition is a string of fixed length k over the alphabet $\{0,1,\#\}$ and each message another string of fixed length k over the alphabet $\{0,1\}$. The sequence of operations are described in Table 1.

Step	*Operation*
1.	A codified message of length k arrives from the environment through the input interface.
2.	Clear the message list.
3.	The message is set in the message list.
4.	All the classifiers that match with some message of the message list are fired. Several rules could be activated in parallel by a message. Don't care symbols, "#", match both 0 as 1.
5.	When a condition is satisfied, the message is posted to the message list.
6.	Step 4 and 5 are repeated for n internal cycles.
7.	Finally, a message is chosen to give the output through the correspondent interface.

Table 1. Sequence of Operations in a Traditional Classifier System.

The system uses a reinforcement algorithm (called the Bucket Brigade, BB, [7]) to solve the credit assignment problem: how to reinforce individual rules in a multistep chain when the external reward is given only at the chain conclusion. This algorithm also allows selection among incompatible or contradictory solutions. BB assigns to each rule a value, called *strength*, that indicates the rule usefulness to the systems goal. When a classifier is matched, it is qualified to participate in an activation auction. To participate in the auction, a classifier makes a bid, proportional to its strength and its specificity (this value is concerned with the number of don't care symbols in the rule). Winning classifiers pay a portion of their strength (their bid) to the one responsible of their activation, and their messages are post to the message list

Finally, a discovery algorithm, a genetic one, is used to generate new, and possibly better, rules into the system. From a CS, a set of rules with higher strength values are selected, genetic operators are applied and the new rules obtained are set into the new CS. After this, the BB will reorganize the rules rate.

A reactive robotic system obtains a new output for each new environmental information sending by the sensors. In this way, it could be defined a decision cycle in a generic robot as it is shown in Table 2.

Step	*Operation*
1.	Read the sensors.
2.	Codify the sensor information to obtain inputs for the controller.
3.	Apply the rules over the inputs to obtain a new output.
4.	Decodify the output in numerical values.
5.	Write the numerical values over the motors.
6.	Go to step 1.

Table 2. Sequence of Operations of a Decision Cycle in a Reactive System.

This process fixed the time range of reacting to environmental changes. The sequence of operations in a traditional CS only consider a new message, as it is shown in the step 1 of Table 1, from this point all the decisions are taken internally without new environmental information.

3. REACTIVE CLASSIFIER SYSTEM (RCS)

The necessity of reacting leads the search of a new mechanism in Classifier Systems that allows to include new environmental codified message in each internal cycle of the performance level. The sequence of operations of this new CS, Reactive CS (RCS), are shown in Table 3 and Figure 2.

Authorized licensed use limited to: Univ Carlos III. Downloaded on March 26, 2009 at 11:41 from IEEE Xplore. Restrictions apply.

Step	*Operation*
1.	A codified message arrives from the environment through the input interface.
2.	The environmental message is fused with messages of previously activated rules.
3.	The environmental message without fusion is also posted to the message list.
4.	All the classifiers that match with some message of the message list are fired.
5.	All the messages of fired classifiers are posted to the message list.
6.	A message is chosen among the rules that satisfied the conditions.

Table 3. Sequence of Operations in a Decision Cycle in a Reactive Classifier System.

This sequence of operations is related with the action level. The Credit Assignment level will be the one which decides what activated rules win in the competition, in the same way that it occurs in the traditional CS.

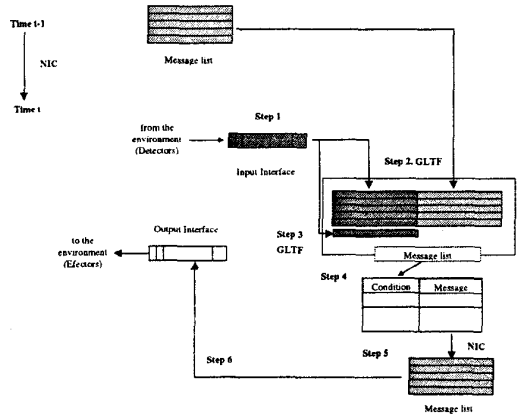

Figure 2. Sequence of Operations in a Decision Cycle in a Reactive Classifier System, Graphically.

This sequence presents two main differences with the traditional CS:

- Generation of message list through fusion, GLTF: The step 2 and 3 in the traditional CS, "clear the message list" and "the codified environmental message is posted to the message list," are translated in two new operations in the RCS: step 2, named "fusion the new message with previous messages", and step 3, "post a new message".
- No internal cycles, NIC: The step 6 in the traditional CS, "the repetition of the step 4 and 5 for *n* internal cycles", is not necessary because the chaining of the rules is performed in each cycle of the performance level.

The loss of internal cycles breaks the rules sequence so characteristic of the traditional CS. To permit rules chaining the codification of the rules in the RCS has been modified. It has been included additional information related to the rules fired in previous instant, this new information is named as internal tags, IT.

The RCS modifies the performance level in order to develop a system able to react (considering only the sensorial input information) and to chain actions (considering information of the sensorial input and the previous state of the CS). These mechanisms instead of the traditional CS, allow the generation of more complex high level rules that are needed for the final solution of the problem. An example of Condition/Action rules that can evolve is as follow:

```
IF External_Signal IS <type 1>        AND
   Last_Rule_Fired IS <type y>
THEN Send_Message <001...>
```

The reaction mechanism, on the other hand, allows the evolution of traditional reaction rules as:

```
IF External_Signal IS <type 1>
THEN Send_Message <001...>
```

Then, action chaining is obtained taking into account two special mechanisms in conditions and messages: the environmental message is fused with the previous messages (GLTF and NIC) and internal tags (IT) are added to evolve a chaining strategy. Reactions are obtained posting a message, with the environmental information only, to the message list.

4. RCS IN ROBOTICS SYSTEMS

To adapt RCS to the navigation problem, conditions and messages of the RCS described in the previous section are divided in three parts. The environmental part of conditions and the decision velocity part of messages concerns with robot state.

The environmental part should be matched with the environmental message arriving from the robot and it is defined by codified sensor values. The environmental message includes all the codified sensors, composed as in Figure 3a. The first part of the message is composed by the proximity sensors to describe the near environment surrounding the robot. The second part corresponds to the goal description using the angle and distance information. The last part of the message deals with the actual velocity to consider the difference between the real and the last decision velocity.

The decision velocity is codified in the output message. Velocity values are decodified and applied to each wheel in the robot, Figure 3b.

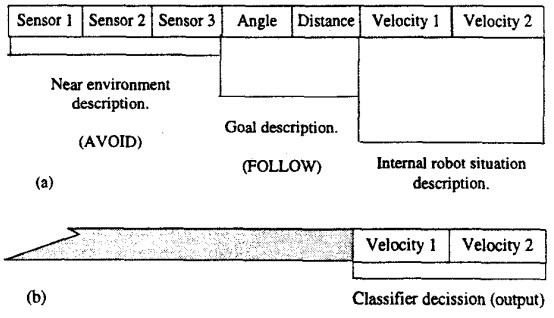

Figure 3. (a) Composition of the environmental message, (b) Decision velocities in the output message.

Authorized licensed use limited to: Univ Carlos III. Downloaded on March 26, 2009 at 11:41 from IEEE Xplore. Restrictions apply.

Sensors description

The robot used is a mini-robot Khepera [10], which is a commercial robot developed at LAMI (EPFL, Laussanne Switzerland). The robot characteristics are: 5.5 cm of diameter in circular shape, 3 cm of height and 70 gr of weight. The sensory inputs come in from eight infra-red proximity sensors. These sensors are composed by two parts: an IR emitter and a receiver. The emitter and the receiver are independent, so that it is possible to use the receiver to measure the reflected light (with the emitter active) or to measure the environmental light (without emission). The reflected light measurement can give some information about the obstacles surrounding the robot. In fact, this measure is not only a function of the distance of an object in front of the emitter but also the environmental light and the object nature (color and texture). So the value of distance is modified by the measure of the ambient light and the object nature, the light use is constant and all the obstacles used have the same color and texture. The robot has two wheels controlled by two independent DC motors with incremental encoder that allows any type of movement. Each wheel velocity could be read by an odometer.

By using the ambient sensors, it is possible to measure the distance and the angle to a light source. The distribution of the amount of light incoming into the eight sensors is used to evaluate the distance and the angle to the source (Figure 4).

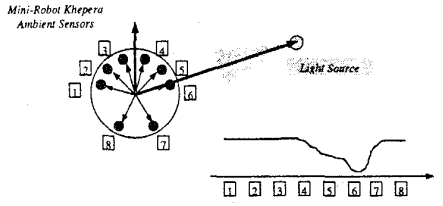

Figure 4: Light Incoming Distribution in the sensors

The transformation function of the proximity sensors is a linear function between (0,0) and (1023,40). The first point corresponds to the minimum distance, 0, both in the real robot as in the codified domain. The second point corresponds with the maximum value, 1023, in the real robot and the codified domain, 40. The function, that transform the ambient sensors values into distance an angle values, searches the minimum intensity value of the proximity sensors and the ID of that sensor. This intensity value is transformed by means of a linear function to get the distance. The desired angle is obtained considering the ID of the sensor with the minimum intensity valued found previously:

Sensor ID	1	2	3	4	5	6	7	8
Angle	90	45	15	345	315	270	190	170

Codification

The sensors (proximity, ambient and odometer) supply three kinds of incoming information: proximity to the obstacles, ambient light and velocity. Instead of using the eight infra-red sensors individually, they have been grouped giving a unique value from two sensor input values, reducing the amount of information received by the RCS. Representing the goal by a light source, the ambient information lets the robot know the angle (the angle position in the robot of the ambient sensor receiving more light) and the distance (the amount of light in the sensor) to this goal (Figure 5).

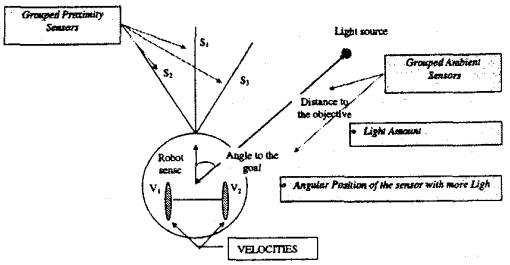

Figure 5: Input information to the system.

The input to the CS consists of three proximity sensors, angle and goal distance (given by ambient sensors) and velocity values obtained by the odometer.

The distance information of proximity sensors is obtained by the response curve of the sensors, that is a sigmoidal function defined over the intensity values domain. The distance domain is transformed, translating it into a more simple domain to codify the values. This transformation allows both the RCS and the robot to be independent. So the RCS could be developed for any robot by changing the transformation function. The input domain has been partitioned in four crisp sets. The maximum distance value "see" by one sensor is 40 units and is divided in ranges as is shown in figure 6.

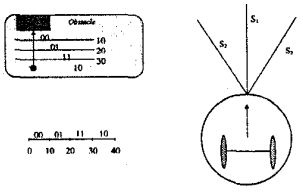

Figure 6: Codification and partition of the proximity information.

The angle sets are of different size to consider a fine fitting of the trajectory, avoiding big oscillations when the robot follows the right direction (the sets near 0 and 2π are smaller than the "$<\pi$" and the "$>\pi$" ones). The input domain partitions are presented in Figure 7.

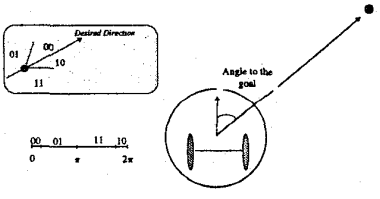

Figure 7: Codification and partition of the angle information.

To keep the independence between robot and RCS, the distance values are translated from the real sensor values to a domain defined from 0 to ∞. The input domain has been partitioned in four crisp sets as is shown in figure 8.

Authorized licensed use limited to: Univ Carlos III. Downloaded on March 26, 2009 at 11:41 from IEEE Xplore. Restrictions apply.

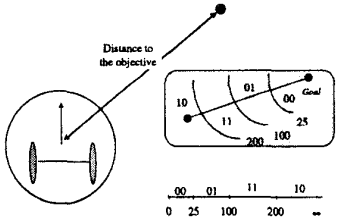

Figure 8: Codification and partition of the distance information.

Velocity values flow as input to the classifier system and as decision from the CS to the robot. The values are defined by the maximum and minimum velocities (10, -10). This range is divided in four equal sets as is shown in figure 9.

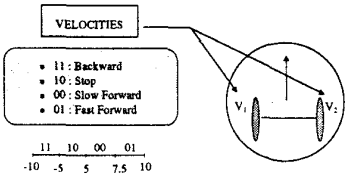

Figure 9: Codification and partition of the velocity information.

All these sets should be codified to build the message from the environment. Two binary digits are needed to represent each set.

5. EXPERIMENTS

Evolution takes a long time of continuous functioning of the hardware. In order to prove the different configurations of the RCS, a simulator developed in a previous work [13] has been used. In the simulator, the characteristics of the turtle robot model [9] and the physical restrictions of the Khepera robot have been considered.

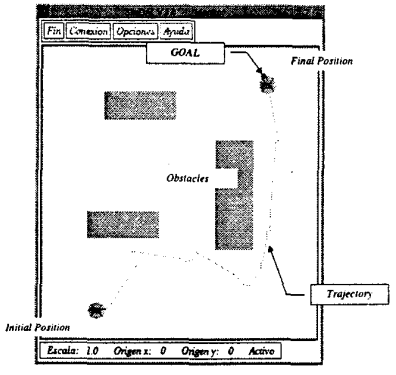

Figure 10: SimDAI Simulator.

The simulation world consists of a rectangular map of user defined dimensions where particular objects are located. In this world it is possible to define a final position for the robot. In this case the robot is represented with three proximity sensors and two special sensors to measure the distance and the angle to the goal (Figure 10).

Different simulated worlds which resemble the real ones have been defined in order to tune the payoff from the environment before being implemented in the real world. An example of these environments can be seen in figure 10. The system developed is the same in both cases (simulated and real) except the differences in the treatment of the sensors, by the transformation function.

Experimental Results

In these experiments, the initial population of the RCS is randomly generated. In this case, it can be proved the ability and improvement of the RCS to learn reactions compared with the traditional approach. The parameters of the CS: traditional and RCS one, are equal:

- The GA is called after 100 cycles of decisions.
- 1 of crossover probability
- 0.01 of mutation probability
- 0.3 of overlapping

Four internal cycles in the performing level are considered in the traditional CS. The simulator executes the robot controller like in the real world, so, while traditional and RCS take a decision the robot is continuously working. The velocity of the robot in this period is the previously decided velocity. This velocity is changed when the CS takes a decision for the incoming environmental message. This consideration takes a main place in traditional CS because it executes four internal cycles before taking a decision.

Figure 11 shows the evolution of the evaluation parameters for the two types of classifiers. In Figure 11 (c) a function that linearly combines the two parameters is shown, the function is: 1,5*time + distance.

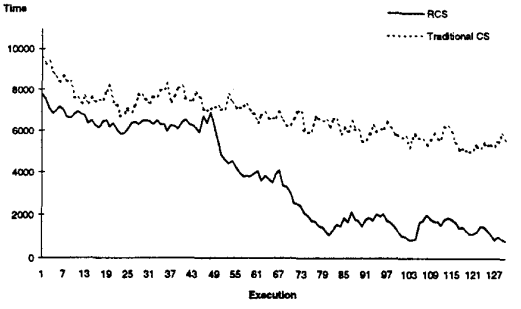

Figure 11.(a): Time to reach the goal by the RCS and the traditional CS.

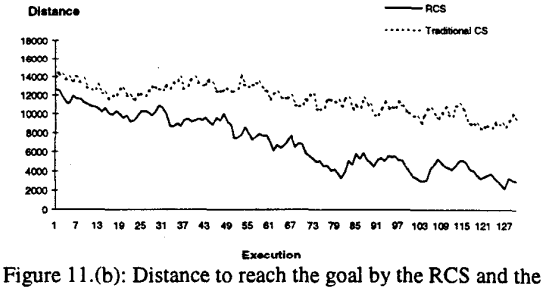

Figure 11.(b): Distance to reach the goal by the RCS and the traditional CS.

Authorized licensed use limited to: Univ Carlos III. Downloaded on March 26, 2009 at 11:41 from IEEE Xplore. Restrictions apply.

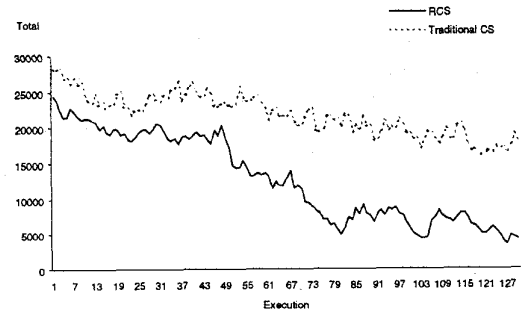

Figure 11.(c): Global evaluation of the two systems.

Figures 11 (a), (b) and (c) show the results of 50 experiments. In these experiments, the seed to create the populations is changed in each generation, therefore, each experiment is related with a different set of initial rules. In the x axis executions are represented. An execution is the navigation of the robot from the initial situation until the goal is reached. In the y axis the average value over 50 experiments of the measured variable is represented both for CS and RCS.

The Figure 11(a) shows the time in finding the goal, it could be seen how the rule learning process causes that the robot finds faster the goal in both of the cases. However, while the traditional CS causes a decrease of about 30%, the RCS could reach a 70% of reduction. This is due to the fact that the RCS is able to learn rules that will be fired just in time, because of the lack of delay between a rule execution and its reward from the environment. The learning of valid chained rules makes the RCS to go faster and straighter to the goal than CS does.

The improvements of the RCS over the traditional CS could be seen in figure 11(c), where the effects of two measures are combined. The achieved rules in RCS improve the performance of the robot in a 60% compared with the rules obtained with the traditional CS.

5. CONCLUSIONS

The proposed RCS has been developed to learn reactions (decision is function of the environmental information) and actions (decision is function of the environmental information and previous internal information). This modified Classifier System has proven its ability to learn autonomous robot behaviors in dynamic environments.

The fusion of each environmental message with information of previous fired rules (GLTF mechanism) and the inclusion of internal conditions (IT mechanism) allow the generation of a sequence of actions, defined by a rule chain over different inputs. Sets of cooperative rules emerge from the evolution of the RCS. Cooperation is viewed in this case as a rules chain, where a rule is only meaningful if it matches with the environment and follows an other specific rule in time.

The inclusion of a message without another information but the environmental input allows the evolution of reactions. The results obtained consider generation of new rules proved the capability of generating not only new better rules but the mechanisms for chaining new and existing rules.

Another important aspect verified in this work is the possibility of continuously learning and adaptation to new situations that allow to solve the problem even if there are mobile objects, more than one goal, and dynamical goals that could appear and disappear or move when the robot is navigating.

REFERENCES

[1] Brooker L., Goldberg D. and J. Holland, "Classifier Systems and Genetic Algorithms". Artificial Intelligence, 40, 1-3, 235-282, (1989).

[2] M. Dorigo, "ALECSYS and the AutonoMouse: Learning to Control a Real Robot by Distributed Classifier Systems", Machine Learning, 19, 209-240, (1995).

[3] Goldberg D.E. "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, New York, (1989)

[4] Holland J., "Adaptation in Natural and Artificial Systems". Ann Arbor, MI, University of Michigan Press, (1975).

[5] Holland J. and Reitman J.S. "Cognitive Systems Based on Adaptative Algorithms", in Waterman D.A. & Hayes-Roth F. (Eds.), Pattern-Directed Inference Systems, New York, Academic Press, (1978).

[6] Holland J. "Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems" in Michasky R., Carbonell J. and Mitchell T.(eds.), Machine Learning: An Artificial Intelligence Approach, vol 2, Morgan Kaufman, los Altos, CA, (1986).

[7] Holland J., "Properties of the Bucket Brigade", Proceedings of an International Conference on Genetic Algorithms and their Applications. Grefenstette J.J., Eds. (1986).

[8] Holland, J.H. "Hidden order : how adaptation builds complexity". Reading (Massachusetts), Addison-Wesley, [1995]

[9] McKerrow P.J., "Introduction to robotics", Addison-Wesley Publishing Company Inc. (1991)

[10] Mondada F. and Franzi P.I. "Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms". Proceedings of the Second International Conference on Fuzzy Systems. San Francisco, USA, (1993).

[11] Sanchis A., Molina J.M., Isasi P., "Classifier Systems for Learning Reactions in Robotics Systems". The first International Workshop on Machine Learning, Forecasting and Optimization (MALFO'96). pp 153-159. Leganés, Spain, July 1996.

[12] Sanchis A., Molina J.M., Isasi P., "Learning Reactive Behavior for Autonomous Robots using Classifier Systems". International Workshop on Spatio-Temporal Models in Biological and Artificial Systems. Sintra, Portugal, November 1996.

[13] Sommaruga L., Merino I., Matellán V and Molina J. "A Distributed Simulator for Intelligent Autonomous Robots", Fourth International Symposium on Intelligent Robotic Systems-SIRS96, Lisboa (Portugal), (1996).

[14] T.H. Westerdale, "Classifier Systems: No Wonder They don't Work", Genetic Programming 1997: Proc. of the Second Annual Conference, 529-537, (1997).

[15] Wilson S.W. "Classifier Systems and the Animat Problem", Machine Learning, 2, 199-228, (1987).

[16] Wilson S.W. "Knowledge Growth in Artificial Animal". Proceedings of an International Conference on Genetic Algorithms and their Applications, (1985)

Authorized licensed use limited to: Univ Carlos III. Downloaded on March 26, 2009 at 11:41 from IEEE Xplore. Restrictions apply.