LEARNING BEHAVIOUR FUSION IN AUTONOMOUS MOBILE ROBOTS*

Antonio Gómez Skarmeta **Humberto Martínez Barberá** **Manuel Sánchez Alonso**

Dep. de Informática, Inteligencia Artificial, y Electrónica
Universidad de Murcia
30071 Spain

skarmeta@dif.um.es, humberto@fcu.um.es

Dep. de Ingeniería y Tecnología de Computadores
Universidad de Murcia
3071 Spain
msanchez@dif.um.es

Abstract

This paper describes the work we are carrying on with intelligent agents for mobile autonomous robots, focusing on the learning procedure for behaviour fusion. We have developed a simulation environment where a robot is modeled as the combination of agents that implement some behaviours. For each agent the behaviours it implements are fused by means of fuzzy rule bases. We show how we apply GAs for the learning of those fusion rule bases.

Keywords: fuzzy control, behaviour learning, behaviour fusion

1 INTRODUCTION

Mobile robots are increasingly required to navigate and perform purposeful autonomous tasks in more complex domains, where the environment is uncertain and dynamic. These requirements demand reactive capacity in their navigation systems. Over the last decade much of the work on reactive navigation has been inspired by the layered control system of the subsumption architecture [4], which tightly couples sensing and action. The emergent behavior of the robot is the result of the cooperation of independent reactive modules, each one specialized in a particular basic behaviour.

Behaviour-based control shows potentialities for robot navigation environment since it does not need building an exact world model and complex reasoning process. However much effort should be made to solve aspects like formulation of behaviours and the efficient coordination of conflicts and competition among multiple behaviors. In order to overcome these deficiencies, some fuzzy-logic based behaviours control schemes have been proposed [13]. Although, most of the behaviour-based approaches focus on static behaviour configurations, introducing the fuzzy rules by means of an expert.

In recent years, fuzzy modeling, as a complement to the conventional modeling techniques, has become an active research topic and found successful applications in many areas. However, most present fuzzy models have been built based only on operator's experience and knowledge, but when a process is complex there may not be an expert [14]. In this kind of situation the use of learning techniques is of fundamental importance. The problem can be stated as follows. Given a set of data for which we presume some functional dependency, the question arises whether there is a suitable methodology to derive (fuzzy) rules from these data that characterize the unknown function as precise as possible. Recently, several approaches have been proposed for automatically generating fuzzy if-then rules from numerical data without domain experts.

In this paper, in order to improve navigation performance in unknown environments, we present a fuzzy-genetic hybrid system, where we combine fuzzy controllers for the different behaviours with a meta controller for fusing the different types of behaviours. This controller is learned using a genetic algorithm. In comparison with traditional approaches the meta-fuzzy controller fuses the behaviours using weights that are learned, instead of simply inhibiting some types of behaviour according to an assigned priority.

2 AGENTS ARCHITECTURE

To assist in the software programming process we have defined a fuzzy logic based language called BG [9]. The user defines agents and their behaviours in BG and then simulates them with a model of the real robot on a given topological map. Once the simulated robot accomplishes the required task the BG program is transferred to the real robot for final testing. BG is based on the C language syntax and semantics and some COLBERT language features [12]. It is based on the multi-agent paradigm using a blackboard [11], where each agent implements a series of behaviours, which make extensive use of fuzzy rules.

Using the BG language and its implicit architecture we have made a decomposition of a real system, extracting the most important elements [1], and defined a series of agents. The following agents have been identified and will be described in the next sections:

* This work has been founded by CICYT TIC97-1343-C002-02 and CICYT-FEDER 1FD97-0255-C003-01 projects

- *Reactive Control*. This is the agent that is in charge of the reactive navigation.
- *Planning*. This is the agent that is in charge of high level planning and global goal completion supervision.
- *Localization*. This is the agent that is in charge of building a model of the environment, and the localization of the robot inside it.
- *Vehicle Control*. This is the agent that is in charge of low level elements control and filtering: sensors and actuators.

2.1 REACTIVE CONTROL

This is the most important agent. It implements the basic reactive behaviours [9], using MISO and MIMO fuzzy rule bases. These rules have been obtained both from previous experience and the literature [8][10].

- *Obstacle-avoider*: drives the robot to the opposite direction to the nearest obstacle.
- *Align-right*: holds the robot at a given distance from the wall at its right.
- *Align-left*: holds the robot at a given distance from the wall at its left.
- *Move-to-goal*: drives the robot in the direction of the current goal.
- *Noise*: drives the robot in a random direction to escape from a hole.

2.2 VEHICLE CONTROL

This agent is composed by two kinds of behaviours: sensor related and actuator related ones. The firsts are in charge of processing and filtering [5] the information provided by the sensors, and the seconds are in charge of sending the right signals to the actuators to perform the given control commands. In our system we have the following behaviours:

- Range sensors acquisition and filtering.
- Collision detection.
- Robot control vector (v, θ) execution.

2.3 PLANNING

Due to the simplicity of the tasks the robot is going to perform, the planning agent [9] is quite simple. At present, it is a deterministic finite state automata, whose states are: looking for object, grasping object, going back to home location. These states have associated goal co-ordinates that will be received by the reactive control agent, which will activate the corresponding behaviours to accomplish that.

2.4 LOCALIZATION

The localization agent [9] is simply in charge of making a map of the environment and keeping track of the home location. To implement the map, one behaviour uses the grid structure provided by the BG language. On the map, the robot is located using odometry (at this time we assume it is ideal, without errors).

3 LEARNING BEHAVIOUR FUSION

A key issue of behaviour-based control is how to efficiently coordinate conflicts and competition among different types of behaviour to achieve a good performance. Instead of just inhibiting and suppressing strategies according to assigned priorities, a fusion method (called behaviour blending) is used [6][9]. A fuzzy rule base carries on this fusion, enabling/disabling the output of the different behaviours, both totally and partially [13][2]. The use of learning techniques [3][6] in the fusion rule base, can result in robot navigation performance improvement. This way, the system can be trained with only some simple behaviours, and then new behaviours can be added without modifying the previous ones: only the fusion rule base needs to be re-trained.

Genetic Algorithms [7] are adaptive procedures of optimization and search that find solutions to problems, inspired by the mechanisms of natural evolution. They imitate, on an abstract level, biological principles such as a population based approach, the inheritance of information, the variation of information via crossover/mutation, and the selection of individuals based on fitness.

GAs start with an initial set (population) of alternative solutions (individuals) for the given problem, which are evaluated in terms of solution quality (fitness). Then, the operators of selection, replication and variation are applied to obtain new individuals (offspring) that constitute a new population. The interplay of selection, replication and variation of the fitness leads to solutions of increasing quality over the course of many iterations (generations). When finally a termination criterion is met, such as a maximum number of generations, the search process is terminated and the final solution is shown as output.

As our main concern is the learning of the fusion rule base, we apply GAs for the task [10]. The user selects an agent, which will serve as the base of the learning procedure. The key idea is to start with a predefined set of fuzzy rules for behaviour blending, and then to apply a learning method to them (rule tuning mode). This set can even be the empty set (rule discovery mode). In either case, the individuals of the GA are fuzzy rules bases, and the result of the GA is the proposed fusion fuzzy rule base.

3.1 INITIAL POPULATION AND CODING

If the algorithm is run in the rule discovery mode the initial population is set randomly. Otherwise, the initial population will consist of the user-defined rules and

random modifications of them. The later corresponds to a typical set up, because it narrows the search space.
The representation we use for the individuals imposes some constraints to the rule bases: the fuzzy sets must be trapezoidal. This is not really a hard constraint because the conversion from triangular to trapezoidal is straightforward. We extract the fuzzy rule base from the blender of a fixed given agent. The user also specifies the different input variables that will be used in the rules. The output variables are the different behaviours of that agent. Each rule is coded, using real numbers, by concatenating the points that define the fuzzy sets of those input and output variables, plus two parameters: the first one indicates if the rule is a regular or a background rule, and the second indicates if the rule is active or not. The representation of the individual is the concatenation of all the different rules (both active or not). For this reason the user additionally establishes the maximum number of rules allowed.

3.2 GENETIC OPERATORS

The user must set up a series of parameters [10] that control how the algorithm proceeds. Basically, which kind of genetic operator is going to be used, the probabilities of both mutation (*Pm*) and crossover (*Pc*), the maximum number of generations (*Geners*) and the population size (*Popul*).
We have implemented several [10] genetic operators:

- Classical mutation
- Non uniform mutation
- Classical crossover
- Arithmetical crossover
- MaxMin crossover

Each time the algorithm needs to apply crossover or mutation, randomly selects one of the previous methods. Additionally the user may specify different probabilities for the different operators. In our examples we have kept all the operators with the same probability.

3.3 OBJECTIVE FUNCTION

While the other topics are quite standard, the definition of the objective function is not straightforward. This function measures the goodness of each individual. As the individuals, in fact, affect the performance of the robot, the objective function must take into account how the robot executes a predefined task. For this task the user must define a simulation framework with the following properties:

- *Goal*: where the robot starts, where the robot ends, and what the robot has to do.
- *Map*: a corridor with walls and obstacles.
- *Iters*: number of simulation runs per individual.

To avoid infinite loops in the simulation some additional parameters are needed:

- *Timeout*: maximum allowable time for each run, measured in epochs.
- *Maxcolls*: maximum allowable number of collisions.

When running the simulation, the current individual is used as the fusion fuzzy rule base of a given agent. Then each simulation run returns four values:

- *Gotcha*: -1.0 if the goal is accomplished, 1.0 otherwise.
- *Epochs*: number of simulations steps performed.
- *Colls*: number of robot-wall collisions.
- *Way*: percentage of distance left from the starting location to the goal one.

Using the different parameters and values defined above, the fitness function is calculated using the following formulae:

$$Total = \sqrt{(start_x - goal_x)^2 + (start_y - goal_y)^2}$$

$$ToGoal = \sqrt{(robot_x - goal_x)^2 + (robot_y - goal_y)^2}$$

$$Way = \frac{Total \times 100}{ToGoal}$$

$$Fitness = \sum_{i}^{Iters} \left[(Timeout_i - Epochs_i) - Colls \times 25 - Gotcha \times 100 - Way \times 5 \right]$$

3.4 SIMULATION RESULTS

We have developed, using the Java language, an integrated development environment for the BG language. This way we have a multi-platform application named BGen (available at *http://ants.dif.um.es/~humberto/asy*). This implements an interpreter for the BG language, a simulator for our custom robot, some data visualization and recording tools, and the GA based learning tool. It is important to note that the robot executes the BGen software, but with a modified version of the simulator that reads in sensory data and writes out the actuator commands. This way both the development and exploitation environment share code. The BGen output includes a representation of the grid map that the robot builds (and the degree of activation of each behaviour as a result of the fusion mechanism. Both are updated in each simulation step.
Using this software we have performed some test to verify the validity of our theses. We will focus on the results of one of such tests. Basically, the goal is to navigate on a simple floor plant (Figure 2), starting from a given location (the left of the arena) and finishing in the opposite side (small circle on the right of the arena). The robot has no idea about the floor plant but the coordinates of the finishing point. The agents, behaviours, and control rules for the robot controller are the ones described and showed in the previous sections. We have fed the learning tool with the Reactive Control agent and the following GA parameters (Table 1):

Table 1. Learning parameters.

Parameter	**Pc**	**Pm**	**Gens**	**Pop**	**Iters**	**Tout**	**Maxcolls**
Value	0.8	0.7	24	4	1	500	5

Then we have obtained the following results. Using the user defined fusion rules, the initial robot controller neither achieves the goal nor collides. This is a good starting point for learning because the controller is pretty fair and intuitive (in fact designed on the fly just a while before the tests), and the learning time should not be very high. As shown below (Figure 1), the fitness function starts with negative values (goal not achieved) and ends up with high positive values (goal achieved in a small number of epochs, near the optimum). The best individual does drive into the goal (Figure 2) despite the fact of the surrounding walls, making tighter turns and minimizing the total distance traveled.

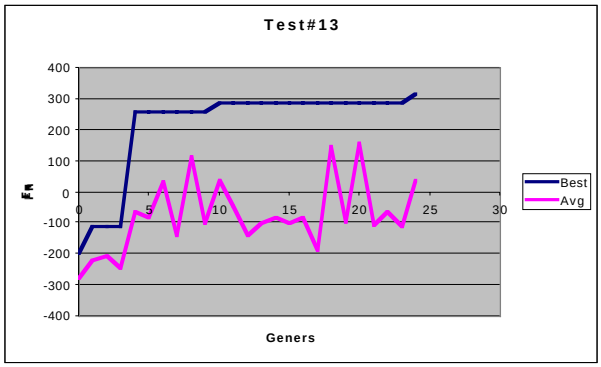

Figure 1. *Fitness* function vs *Geners*.

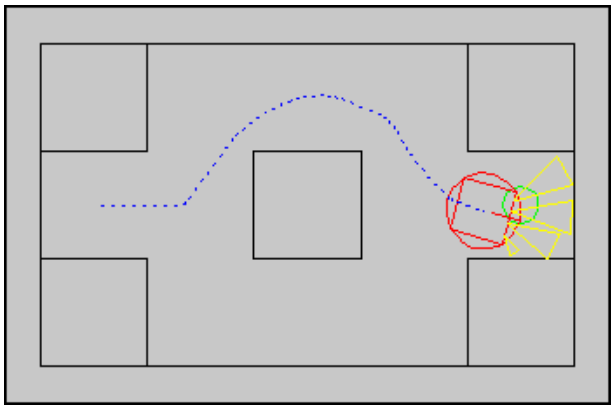

Figure 2: Simulation result after learning.

4 CONCLUSIONS

In this paper a hybrid fuzzy-genetic agent based system for autonomous robots in uncertain environments is proposed. These agents are composed of two components: first a collection of fuzzy controllers that implement the different behaviours-based schemes of the robot, and second, a fuzzy meta-controller that combines the different behaviours and where the weights associated to each behaviour are learned using a genetic algorithm. In this way this meta-controller coordinates conflicts and competition among multiple behavious efficiently. We have shown how this learning is carried on with the agent architecture we have defined. We have developed some tools to assist on the overall controller deployment process. To test the performance of the whole system we have carried out several tests with the simulator. The results are quite satisfactory. Future work includes the test of more complex tasks and environment conditions, and the study of the correlation between the simulator and the real robot performance.

References

[1] D.G. Armstrong, C.D. Crane, "*A Modular, Scalable, Architecture for Intelligent, Autonomous Vehicles*", 3rd IFAC Symposium on Intelligent Autonomous Vehicles, Universidad Carlos III, Madrid, Spain, pp 62-66, 1998

[2] B.C. Arrúe, F. Cuesta, B. Braunstingl, A. Ollero, "*Fuzzy Behaviours Combination to Control a Non Holonomic Mobile Robot using Virtual Perception Memory*", Sixth IEEE Intl. Conf. on Fuzzy Systems (FuzzIEEE'97), Barcelona, Spain, pp 1239-1244, 1997

[3] A. Bonarini, F. Basso, "*Learning to Compose Fuzzy Behaviors for Autonomous Agents*", Intl. Journal of Approximate Reasoning, no. 17, pp 409-432,

[4] R.A. Brooks, "*A robust Layered Control System for a Mobile Robot*", IEEE J. Robotics and Automat., vol. RA-2, no. 1, pp 14-23, 1986

[5] M. Delgado, A.F. Gómez Skarmeta, H. Martínez Barberá, J. Gómez, "*Fuzzy Range Sensor Filtering for Reactive Autonomous Robots*", Fifth Intl. Conference on Soft Computing and Information / Intelligent Systems (IIZUKA'98), Fukuoka, Japan, 1998

[6] M. Dorigo, M. Colombetti, "*Robot Shaping: an Experiment in Behavior Engineering*", The MIT Press, Cambridge, MA, 1998

[7] D.E. Golberg, "*Genetic Algorithms in Search, Optimization, and Machine Learning*", Addison-Wesley, 1989

[8] A.F. Gómez Skarmeta, H. Martínez Barberá, "*Fuzzy Logic Based Intelligent Agents for Reactive Navigation in Autonomous Systems*", Fifth International Conference on Fuzzy Theory and Technology, Raleigh, USA, pp. 125-131, 1997

[9] A.F. Gómez Skarmeta, H. Martínez Barberá, M. Sánchez Alonso, "*A Fuzzy Agents Architecture for Autonomous Mobile Robots*", International Fuzzy Systems World Congress (IFSA'99), Taiwan, 1999

[10] A.F. Gómez Skarmeta, F. Jiménez Barrionuevo, "*Generating and Tuning Fuzzy Rules with Hybrid Systems*", Sixth IEEE Intl. Conference on Fuzzy Systems (FuzzIEEE'97), Barcelona, Spain, pp. 247-252, 1997

[11] B.A. Hayes-Roth, "*Blackboard for Control*", Artificial Intelligence, vol. 26, pp 251-324, 1985

[12] K. Konolige, "*COLBERT: a Language for Reactive Control in Saphira*", German Conference on Artificial Intelligence, Freiburg, 1997

[13] A. Saffiotti, "*Fuzzy Logic in Autonomous Robotics: behavior coordination*", Sixth IEEE Intl. Conference on Fuzzy Systems (FuzzIEEE'97), Barcelona, Spain, pp. 573-578, 1997

[14] L. Wang, R. Langari., "*Complex Systems Modeling via Fuzzy Logic*", IEEE Trans. on Systems, Man and Cybernetics, vol. 26, no. 1, pp. 100-106, 1996