

Fuzzy logic controller based on XML formatted files for behaviour-based mobile robots

Fabio Tampalini and Riccardo Cassinis

Department of Electronics for Automation

University of Brescia

IT - Brescia

{fabio.tampalini,riccardo.cassinis}@ing.unibs.it

Abstract—A fuzzy logic controller for behaviour-based mobile robots is presented. All knowledge is contained in XML formatted files. In this way the fuzzy logic controller is completely independent from the robot. After having described the philosophy and the structure of the system, we show the results of experiments performed using the simulator we have created.

Index Terms—Fuzzy logic, XML, behaviour, inferential engine

I. INTRODUCTION

The state of the art of mobile robots shows how the use of robots to support human activities is more and more aimed to specific and complex tasks.

This requires robots equipped with a high number of devices to accomplish needed operations, often resulting in high energy consumption and consequent low autonomy. Such robots are also expensive and possibly unreliable, due to complexity.

One of the goals for the future of mobile robots seems to be the creation of robot swarms, where units cooperate. In this case, each robot can be equipped just for a specific, simple task; it is easier to replace in the community it works in, and is much cheaper and more robust. Autonomy is also increased and the robot can work for longer periods or move to farther destinations.

In fact in a specific time slice not all robot are needed to work but only those equipped for the current task. In the worst case, when all robots are necessary for the current task, the global energy consumption is higher than in the case of a single machine, but different jobs are executed in parallel, increasing the potentiality of the global system. Another way of reducing complexity and energy consumption is to remove from the physical robot all the devices that do not require being on board: this includes processing power and some sensors. In this case, we can refer to the machine as a “pseudo tele-operated robot”: it simply executes instructions received from a remote controller (or from a cluster of remote controllers). This particularly applies to robot swarms, where removing computing power from robots and placing it elsewhere can lead to significant advantages.

Esteroeceptive and proprioceptive sensors can produce information that are sent to the remote controller. The controller thus has a description of the state of the robot and uses this

information, together with the task to be accomplished, to decide instructions for the robot to execute. Commands to the robot actuators are sent to the swarm’s unit whose only task is to execute them.

This is the purpose of this project: to create a fuzzy inference engine, fully independent from the robotic architecture it will manage, from the environment it will operate in, and from the type and position of the sensors that will be used. We decided to use fuzzy logic because it is robust to noise, and it also allows co-existence of conflicting behaviours whose contradictions are solved through blending.

We decided to interface our engine to the world through XML formatted files: the engine can then be considered fully independent from the way we will use it, and it can coordinate robots with different structures, in various environments and with different tasks. These files store all the information needed to operate: input values, behaviours, rules, etc. Our engine is just an interpreter which operates in a “Babel” of robot architectures [Blank et al., 1999], environments, tasks, sensors, etc.

A block diagram of the inferential engine is shown in Figure 1. XML file format can be considered a *de facto* standard, and we can use a text editor to manage information, or any other structure.

XML was adopted because it is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web [Quin, 2004].

This makes our engine capable of operating easily with existing or even future systems.

In the sequel we will focus on our engine as a purely reactive architecture. To obtain a deliberative or hybrid architecture one can simply change the contents of XML formatted files. The architecture, named Robutt after a robot dog from an Asimov novel, is based on a robocentric system, where all pieces of information to be computed are referred to the actual position of the robot.

II. FUZZY INFERENCE ENGINE

In order to study the functionality of this system, we have tested it in simple robot guidance tasks, like “obstacle-avoiding” and “target-reaching”

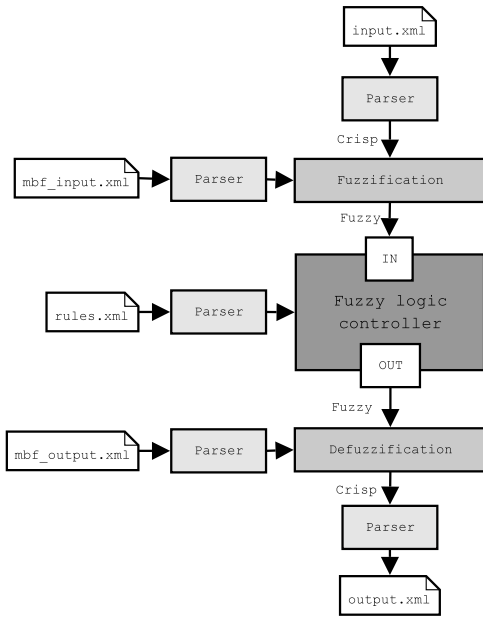


Fig. 1. Global architecture.

A purely reactive system could be a good solution for these tasks, as it is fast, requires few computational resources, and an already existing planner can be used over it. Other advantages of purely reactive systems are:

- emphasis on the importance of a tight relationship between perception and action;
- absence of abstract knowledge and symbolic reasoning;
- vertical decomposition of the problem into sub-problems to be executed in parallel;
- modularity of the software;
- architectures are often inspired by theories from several disciplines (i.e. biology, psychology, neurology).

Following the classics outlines [Brooks, 1986] we could say that Robutt is composed of two components: functional module and behavioural module. The functional component

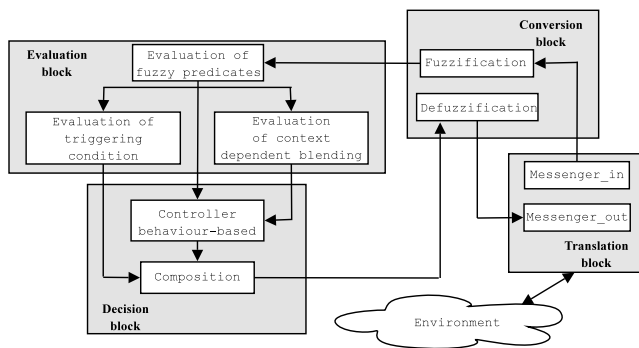


Fig. 2. Functional view.

(Figure 2) acquires information to be used as input to the engine. As these data have been processed, the functional component blends resulting actions and transmits values to actuators. The functional component is composed of:

- translation block: it is an interface between information about the environment surrounding the robot (stored in XML formatted files) and data processed inside the engine;
- conversion layer: information acquired from translation layer are here transformed into fuzzy values (fuzzification) and, after computations, output fuzzy values are re-transformed into crisp values (defuzzification);
- calculus layer: it is composed by three sub-modules, each one managing a sub-tree: predicates sub-tree, behaviour triggering conditions sub-tree and behaviour evaluations sub-tree;
- decision layer: decides actions to be carried out on the basis of environment information that are provided by previous layers. Behaviours are entities totally independent from each other and from the environment, describing activities to be carried out [Invernizzi and Labella, 2000].

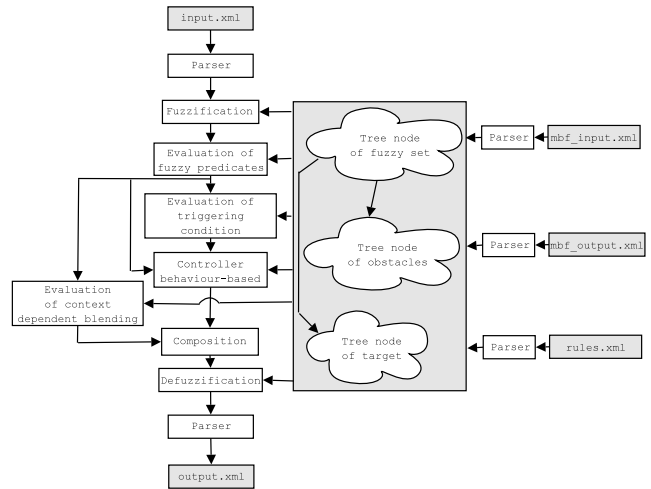


Fig. 3. Structural view.

The structural scheme (Figure 3) shows the proposed architecture and how it is focused on modular development for the system to be easily reusable and adaptable to future necessities.

We chose to use an inferential engine based on fuzzy logic to create our interpretative black box based on XML formatted files. This implies that to attain a given target it is necessary to decompose the problem in sub-behaviours and to blend their outputs to accomplish the global task. Basic behaviours are extremely specialistic: only by blending them we obtain global “intelligent” behaviours. Outputs from basic activated behaviours and from the engine are blended by the Blending Module. Also this block acts on the basis of the global strategy described in XML formatted files.

III. INPUT/OUTPUT

The engine input is an XML formatted file (Table I). In our test the goal for the robot is to move to a target object avoiding some obstacles. Each node in the XML

```

<?xml version="1.0"?>
<object>
  <target>
    <name>T1</name>
    <distance>350.5</distance>
    <heading>0</heading>
    <radius>5</radius>
  </target>
  <obstacle>
    <name>O1</name>
    <distance>230.5</distance>
    <heading>3</heading>
    <radius>5</radius>
  </obstacle>
  <obstacle>
    <name>O2</name>
    <distance>130.5</distance>
    <heading>40</heading>
    <radius>5</radius>
  </obstacle>
</object>

```

TABLE I

EXAMPLE OF AN XML FORMATTED INPUT FILE DESCRIBING THE ENVIRONMENT SURROUNDING THE ROBOT.

input file describes an object in the environment. The node label is used by the XML parser to identify different entities in the environment: information enclosed between labels `target` and `/target` describe the target object, while sub-trees beginning with `obstacle` labels provide information about objects to be avoided while performing the task.

Our work is not aimed at determining the type and position of objects present in environment. These pieces of information should be generated by external devices.

Variables stored in these nodes describe:

- distance [mm];
- heading [°];

During the testing phase the simulator (Section VI) generates for any loop an input file to describe the new environment state. This file describes the environment after simulated execution of the instruction of the previous loop. The engine output is a new XML formatted file containing information that the robot controller must simply transfer to its actuators. In our experiment we used the robot MARMOT (Mobile Advanced Robot for Multiple Office Tasks, developed at the Advanced Robotics Laboratory of our university), Figure 4.

MARMOT has three omnidirectional wheels, that make it holonomous (Figure 5). An interesting thing about such structure is that, since it has a non-redundant structure, any movement command to its motors results in an actual movement of the robot, and no checking of the consistency of movement commands is required. As the engine receives the input file, it fuzzifies each crisp value on the basis of the contents of the membership function file. This process assigns a credibility degree to each input according to its membership function. After the computations described in the following paragraphs have been completed by the



Fig. 4. MARMOT.

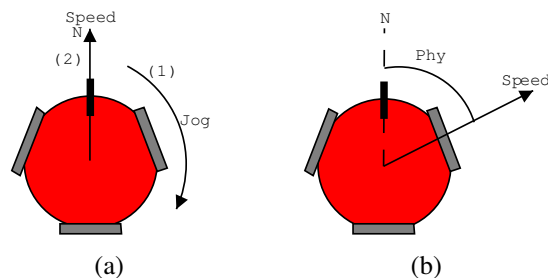


Fig. 5. Representation of two kinds of possible movements.

engine, an opposite process is carried out to transform fuzzy values into crisp ones. Therefore, by modifying the membership function input/output files (Table II), it is possible for the engine to work in any field and with any goal, since these two files are the interfaces between the controller and the world.

IV. FUZZY RULES

Fuzzy rules are the bricks to build the operative knowledge of the robot on the basis of a human expert's heuristic knowledge.

Fuzzy rules template is:

IF $\langle \text{antecedent} \rangle$ THEN $\langle \text{consequent} \rangle$

where the antecedent could consist of an arbitrarily large number of preconditions combined through logic operators OR, AND and NOT; for example:

IF (Target \in N AND Target \in far AND NOT (Obstacle \in N)) THEN (Speed \in mid AND Phy \in N AND Jog \in zero)

This fuzzy rule states that if the target to be reached is far and north of the robot, and no obstacles are in the north direction, then the robot must advance pretty fast along its North direction.

Obviously, while in the antecedent all the aforementioned logic operators can be used, in the consequent only the AND operator is acceptable.

Moreover, credibility values (i.e. the membership degree of a variable to a membership function) range between "0"

```

<?xml version="1.0"?>
<robot_input>
  <distance>
    <close>
      <type>trapeze</type>
      <a>0</a>
      <b>0</b>
      <c>40</c>
      <d>70</d>
    </close>
    <near>...</near>
    <far>...</far>
    <very_far>...</very_far>
  </distance>
  <heading>
    <N>
      <type>untrapeze</type>
      <a>0</a>
      <b>45</b>
      <c>315</c>
      <d>360</d>
    </N>
    <NE>...</NE>
    <E>...</E>
    <SE>...</SE>
    <S>...</S>
    <SW>...</SW>
    <W>...</W>
    <NW>...</NW>
  </heading>
</robot_input>

```

TABLE II

AN EXAMPLE OF XML FORMATTED FILE DESCRIBING INPUT FUZZY SETS.

```

<rule>
  <speed>stop</speed>
  <jog>zero</jog>
  <phy>N</phy>
  <and>
    <target>close</target>
    <target>N</target>
  </and>
</rule>

```

TABLE III

PART OF RULES FILE.

target, avoiding obstacles, etc.), thanks to a set of rules we obtain a “global intelligent behaviour”. Writing a set of rules is not commonplace, it is necessary to adopt strategies and devices resulting from experience and intuition [Saffiotti et al., 1997]. While creating rules we as-

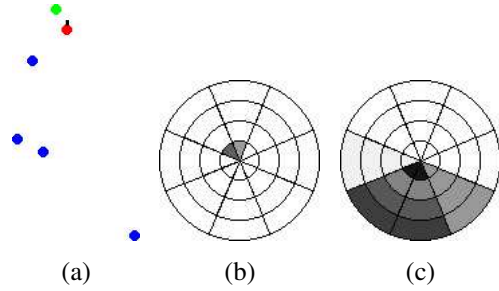


Fig. 6. Representation of the environment surrounding the robot in the graphical interface of the simulator (a); robot’s view of the target position (b); robot’s view of obstacles positions (c).

and “1”, included. This implies that *T-norm* and *T-conorm* are just the AND and OR operators of classic logic, where:

- *T-norm*
 - 1) $\min(x, y)$
 - 2) $x \cdot y$
 - 3) $\max(x + y - 1, 0)$
- *T-conorm*
 - 1) $\max(x, y)$
 - 2) $x + y - x \cdot y$
 - 3) $\min(x + y, 1)$

Rules are transmitted to the inferential engine through the `rules.xml` file that also supplies information to compute fuzzy inferences. Thresholds of triggering conditions and blending rules are also stored in this file. More precisely we can say that the `rules.xml` file contains the robot behaviours: for this reason the file is structured as in Table III.

Table III shows the form of the rule described in the previous example: the `rule` node contains information for the engine to compute the fuzzy inference. As a matter of fact, the credibility value of the `speed` output variable in its membership function is the result obtained through *T-norm* operator whose inputs are the degrees of `target` membership to `close` and `N`; the same real number is attributed to output variables `jog` for `zero` and `phy` for `N`.

Each rule is necessary for a certain task (reaching the

summed that there was only one target and that all the obstacles had the same importance. This led to implementing a target position map and an obstacles one that are scanned when the engine processes the rules antecedents. This can be seen in Figure 6, that shows (in image (a)) the position of the robot, of the target that has just been reached and of the obstacles. The following two images (b) and (c) contain a graphic representation of what has been explained before: we calculate an AND for every membership function couple of *distance* and *heading*, then we calculate an OR between the couples of all the obstacles thus obtained.

V. TRIGGERING CONDITIONS AND CONTEXT DEPENDENT BLENDING

A. Triggering conditions

We introduced in the previous section the concepts of the activation threshold of the behaviour and the outputs of basic behaviours blending (this will be shown in the next subsection) to obtain a functional engine.

One can see the fuzzification, blending and defuzzification blocks in the functional engine scheme (Figure 7). The effective engine component is the one labeled *Inference*. This is the scheme we have chosen for Robutt, in which the block that evaluates the triggering condition is scanned before the sub-tree of behaviour (in order to avoid wasting

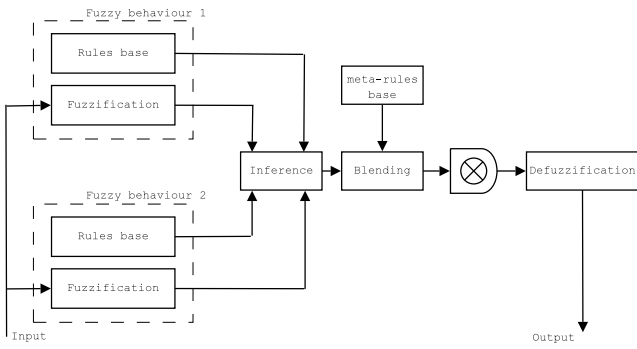


Fig. 7. Fuzzy inferential engine scheme.

computational resources) and there is a blending block for each behaviour. The purpose of activation threshold is to state the effective possibility that the robot behaviour is not going to change.

In this way the computational load is decreased because the engine is not forced to scan all the rules contained in the file.

B. Context dependent blending

Another important component is the blending block that fuses the outputs of basic behaviours. This block allows the coexistence of behaviours even if there are conflicting tasks to be performed.

Various strategies exist for building the blending block, and all belong to one of two large groups: “cooperative” and “competitive”. In the latter case the arbitration is quite simple: only one basic behaviour wins. This strategy has some advantages, but its drawbacks are significant, the main one being the fact that the “fuzzy” idea is completely lost.

For these reasons Robutt uses a cooperative arbitration scheme, more qualitative than quantitative (therefore it is similar to fuzzy logic); in fact the cooperative arbitration allows all activated behaviours to contribute in some way to the controlling action that is output from the engine. The relative weight of each behaviour depends on the kind of arbitration used to manage the behaviour itself. In fact using the competitive arbitration scheme the weights of losing behaviours become useless.

To implement the arbitration we shall use the strategy in which there is for every fuzzy rule a blending block, or the strategy that has a blending block for each basic behaviour.

Blending rules are contained in the component labeled *meta-rules base* (Figure 7)

Formally:

$$Des_{B_i}(\underline{x}, \underline{c}) = \max_k \left\{ \left\{ \min \left[d_1^k * \mu_{A_1}^k(x_1), d_2^k * \mu_{A_2}^k(x_2), \dots, d_N^k * \mu_{A_N}^k(x_N) \right] \right\} \mu_C^k(\underline{c}) \right\}$$

$$Des_{B_i}(\underline{x}, \underline{c}) = \max_k \left\{ \min \left\{ \min \left[\mu_{A_1}^k(x_1), \mu_{A_2}^k(x_2), \dots, \mu_{A_N}^k(x_N) \right], d^k * \mu_C^k(\underline{c}) \right\} \right\}$$

Des_{B_i} - desiderability function [Ruspini, 1991], [Enrique, 1991], [Saffiotti et al., 1999]

$d_2^k, d_1^k, \dots, d_N^k$ - relative weights of k-th meta-rule.

In Robutt we use the strategy that has a blending block for each basic behaviour.

VI. SIMULATIONS AND TESTS

To test the engine, and to create an archive of basic behaviours, we built a simulator with graphical user interface (Figure 8). This simulator is configurable to analyze the robot’s response time.

Tests were run on a laptop Acer 525TE, PIII, 700Mhz and 128MB RAM, with ATI Rage Mobility Pro 2x (8MB).

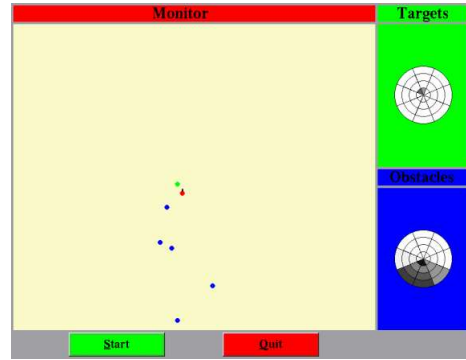


Fig. 8. Graphic User Interface.

The simulator inputs are stored in the XML formatted file as crisp values to be executed by the robot actuators. The simulator then evaluates, given a configurable temporal resolution, the relative displacement of the robot, creates an XML formatted file containing fuzzy values to describe the environment surrounding the robot. This file is fed to the inferential engine.

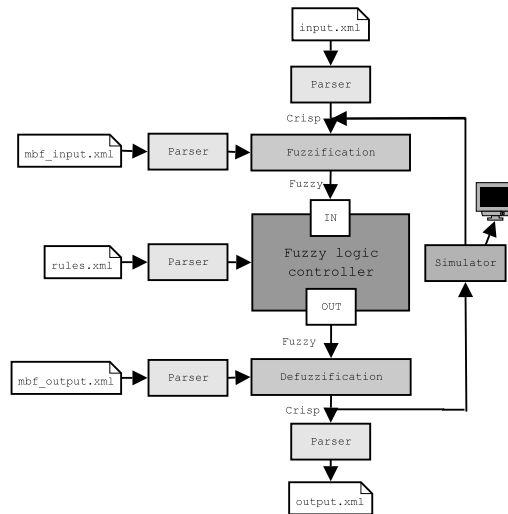


Fig. 9. Global architecture with simulator.

As a first case of study, we chose a simple, yet significant problem: navigation towards a known goal in an unknown,

moderately cluttered 2-D environment. If we transpose the engine in an ethnological field of study, the robot's actions are determined by these behaviours: aversion, protection and pursuit.

Robutt is a purely reactive architecture so it takes decisions step by step. This allows it to work even if obstacles and the target are moving; Robutt takes information about environment as a stipe picture.

To test Robutt we implemented two behaviours:

- reaching-the-target;
- avoiding-obstacles.

We decided that the most important point was the robot safety: therefore the reaching-the-target behaviour weight is lower than the avoiding-obstacles behaviour weight.

The following data were directly measured from the behaviour of the developed system:

- total step execution time;
- parsing time;
- time to create memory lists;
- inference time.

These values were measured in three different environmental conditions: target only, target and five obstacles, target and nine obstacles (Table IV).

	target	target 5 obstacles	target 9 obstacles
Total time	4.9 μs	5.3 μs	5.5 μs
Parsing time	3.5 μs	3.7 μs	3.9 μs
Time to create lists	0.2 μs	0.2 μs	0.3 μs
Inference time	1 μs	1.2 μs	1.2 μs

TABLE IV

RESULTS OBTAINED WITH PURELY REACTIVE ENGINE ONLY.

The target is 9.99 meters from the robot's starting position. Using the simulator we also gathered data about:

- total task execution time;
- parsing time of all input files;
- time to create memory trees;
- inference time.

These were measured in a world with no obstacles, and in one with five obstacles (Table V).

It is important to stress that the robot reaches the target in all tests. The timeout condition (Table V) is forced to determine the in maximum time value in the worst case.

We obtained satisfying results for all computational cycle times and for task completion times even with a large number of objects in the environment where the robot worked.

VII. CONCLUSIONS AND FUTURE RESEARCH

The characteristic of Robutt of being independent from the application domain results in an extremely flexible and configurable engine, thanks to the XML formatted files that contain all the knowledge.

Test results satisfied all our expectations, and we are now in the process of inserting the inferential engine into our

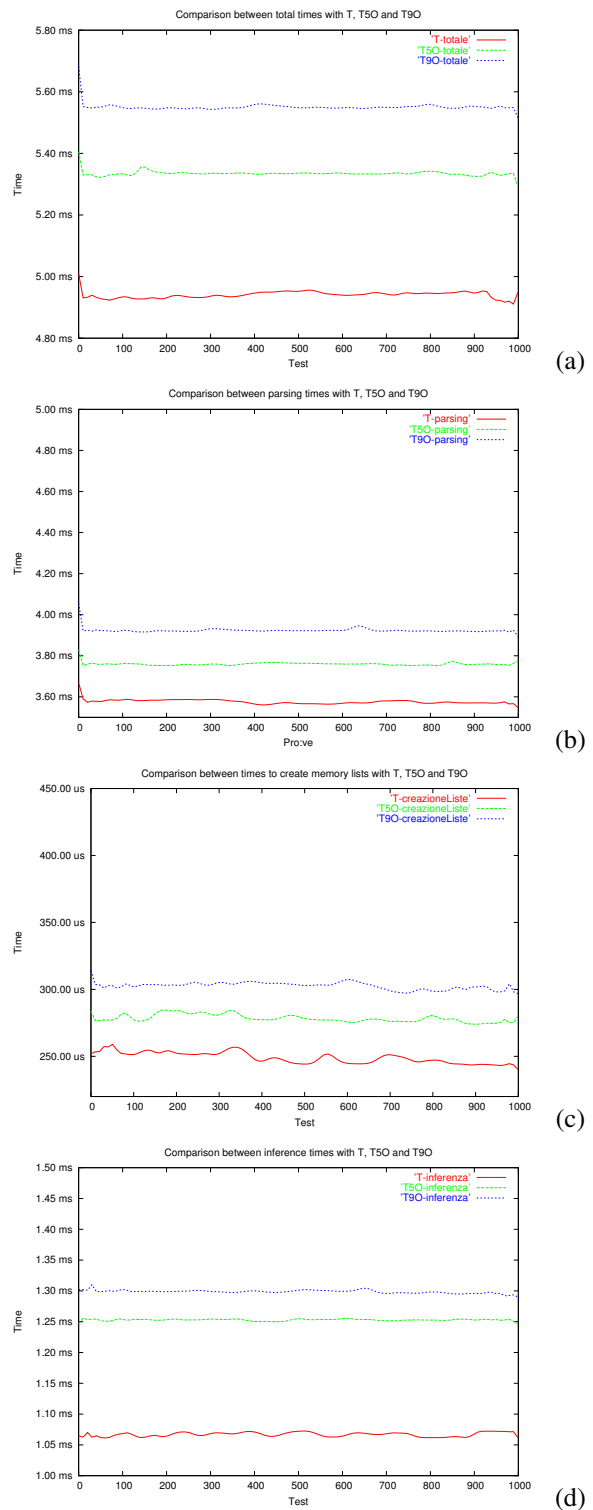


Fig. 10. Graphical representation of tests.

global architecture for robot swarms **Device Communities Development Toolkit** [Cassinis et al., 2001]. In fact in a global system in which robots, calculators, satellites, sensors, cameras, webcams, etc. work, the controller has every kind of data deriving from the exchange of the messages between community members. Robutt has been

Objects	Time slice [s]	Simulator type	Parsing [ms]	Creation lists [ms]	Inference [s]	Graphical interface [ms]	Loop numbers
T	0.01	textual int.	4.2	0.2	1.9432	-	312
T	0.10	textual int.	4.2	0.2	0.1464	-	30
T+50	0.01	textual int.	4.0	0.3	1.7325	-	340
T+50	0.10	textual int.	4.1	0.4	0.1910	-	33
T+50	0.01	textual int.	3.9	0.2	3.2106	-	600 (time-out)
T+50	0.10	textual int.	4.0	0.2	3.8096	-	600 (time-out)
T	-	graphical int.	-	-	-	11.7	1 (forced)
T+50	-	graphical int.	-	-	-	15.6	1 (forced)
T	0.01	graphical int.	3.8	0.2	4.6242	-	312
T	0.10	graphical int.	3.8	0.2	1.6213	-	30
T+50	0.01	graphical int.	3.9	0.2	6.5685	-	340
T+50	0.10	graphical int.	3.9	0.2	1.4345	-	33
T+50	0.01	graphical int.	4.0	0.2	10.4223	-	600 (time-out)
T+50	0.10	graphical int.	3.9	0.2	11.1641	-	600 (time-out)

Initial conditions about target: distance is 999 mm and heading is North.

TABLE V
SIMULATION RESULTS.

realized as a purely reactive architecture, it has no memory and cannot manage any task requiring memory. This resulted in a purely reactive engine [Arkin, 1998], but it is possible to use our system with a deliberative/planner block (e.g. Saphira [Konolige et al., 1997][Konolige, 1999] [Konolige and Myers, 1996]), creating an hybrid architecture, that is perhaps the best solution [Lyons and Hendricks, 1992]. Different architectures are obtained just by setting appropriately XML formatted files, a feature that allows Robutt to be even more highly usable and functional.

REFERENCES

- [Arkin, 1998] Arkin, R. C. (1998). *Behaviour-Base Robotics*. MIT-Press.
- [Blank et al., 1999] Blank, D. S., Hudson, J. H., Mashburn, B. C., and Roberts, E. A. (1999). The XRCL Project: The University of Arkansas' Entry into the AAAI 1999 Mobile Robot Competition. Technical Report CSCE-1999-01, The University of Arkansas. <http://dangermouse.brynmawr.edu/xrcl/>.
- [Brooks, 1986] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14-23. <http://www.ai.mit.edu/people/brooks/papers/AIM-864.ps.Z>.
- [Cassinis et al., 2001] Cassinis, R., Meriggi, P., Bonarini, A., and Matteucci, M. (2001). Device communities development toolkit: An introduction. In *Proceedings of EUROBOT'01*. Lund, Sweden.
- [Enrique, 1991] Enrique, R. (1991). Truth as utility: A conceptual synthesis. In *Proceedings of the 7th Annual Conference on Uncertainty in Artificial Intelligence (UAI-91)*, pages 316-322, San Mateo, CA. Morgan Kaufmann Publishers.
- [Invernizzi and Labella, 2000] Invernizzi, G. and Labella, T. H. (2000). Sviluppo di un gestore di comportamenti fuzzy per agenti autonomi. Master's thesis, Politecnico di Milano.
- [Konolige, 1999] Konolige, K. (1999). *Saphira users' manual*.
- [Konolige and Myers, 1996] Konolige, K. and Myers, K. (1996). *The Saphira Architecture for Autonomous Mobile Robots*. MIT-Press.
- [Konolige et al., 1997] Konolige, K., Myers, K. L., Ruspini, E. H., and Saffiotti, A. (1997). The Saphira architecture: A design for autonomy. *Journal of experimental & theoretical artificial intelligence: JETAI*, 9(1):215-235. <http://iridia.ulb.ac.be/saffiotti/abstracts.html>; <ftp://iridia.ulb.ac.be/pub/saffiotti/robot/jetai96.ps>.
- [Lyons and Hendricks, 1992] Lyons, D. M. and Hendricks, A. J. (1992). Planning, reactive. In *Encyclopedia of Artificial Intelligence*, pages 1171-1181. John Wiley, 2nd edition.
- [Quin, 2004] Quin, L. (2004). Extensible markup language (xml) activity statement. <http://www.w3.org/XML/Activity>.
- [Ruspini, 1991] Ruspini, E. H. (1991). On the semantics of fuzzy logic. *Int. J. Approx. Reasoning*, 5(1):45-88.
- [Saffiotti et al., 1999] Saffiotti, A., Ruspini, E., and Konolige, K. (1999). Using fuzzy logic for mobile robot control. In Zimmermann, H.-J., (Ed.), *Practical Applications of Fuzzy Technologies*, volume 6 of *Handbooks of Fuzzy Sets*, chapter 5, pages 185-205. Kluwer Academic, MA.
- [Saffiotti et al., 1997] Saffiotti, A., Ruspini, E. H., and Konolige, K. (1997). Using fuzzy logic for mobile robot control. In Dubois, H. P. D. and Zimmermann, H. J., (Eds.), *International Handbook of Fuzzy Sets and Possibility Theory*, volume 5. Kluwer Academic Publishers Group, Norwell, MA, USA, and Dordrecht, The Netherlands. Forthcoming.