Real-Time Adaptive Fuzzy Motivations for Evolutionary Behavior Learning by a Mobile Robot

Wolfgang Freund, Tomás Arredondo V., César Muñoz, Nicolás Navarro and Fernando Quirós

Universidad Técnica Federico Santa María, Valparaíso, Chile,
Departamento de Electrónica,
Casilla 110 V, Valparaíso, Chile,
wolfgang.freund@usm.cl

Abstract. In this paper we investigate real-time adaptive extensions of our fuzzy logic based approach for providing biologically based motivations to be used in evolutionary mobile robot learning. The main idea is to introduce active battery level sensors and recharge zones to improve robot behavior for reaching survivability in environment exploration. In order to achieve this goal, we propose an improvement of our previously defined model, as well as a hybrid controller for a mobile robot, combining behavior-based and mission-oriented control mechanism. This method is implemented and tested in action sequence based environment exploration tasks in a Khepera mobile robot simulator. We prove our techniques with several sets of configuration parameters on different scenarios. The experiments shows an significant improvement in robot responsiveness regarding survivability and environment exploration.

Keywords: Fuzzy logic, mobile robot, real-time, environment exploration.

1 Introduction

Real-time systems are concerned with real-world applications, where temporal constraints are part of system specification imposed by the environment, i.e. firm-deadlines in QOS environments, soft-deadlines in non-critical control applications and hard deadlines in safety-critical systems. In the last years more research effort have been made applying soft-computing techniques to real-time control problems [1–5]. The main advantage over traditional control mechanisms is in the additional robustness regarding lack or poor environmental information (if not the problem definition itself) which concern almost all real-time control applications [6].

On the other hand, soft-computing based methods are more intuitive than strict formal models, soft-computing (e.g. fuzzy logic) aim to gain from operator perceptions and through iteration obtain capabilities of the real expert. However,

not much attention has been given to real-time considerations, regarding soft or hard deadlines. Some important aspects of real-time must be taken into account: how could soft-computing techniques, such as fuzzy logic, neural networks or genetic algorithms affect systems responsiveness and survivability?

Recently, we have proposed a fuzzy logic based method that provides a natural interface in order to give a variety of motivations to be used in robotic learning. To test the validity of the proposed method we tested the fuzzy logic based method on behavior based navigation and environment recognition tasks within a Khepera robot simulator [7].

In order to introduce our behavior based mobile robot methodology in a real-world application, we introduce an active battery sensor to allow for the detection of low battery conditions and we also provide various number of recharge zones withing different room configurations. This real-time extension must be capable of supporting different sets of motivations, improving survivability and exploration performance.

Our primary goal consists of full environment exploration considering energy consumption and recharge zones. To reach this target, robot's behavior must be influenced through fitness evaluation for recharging the battery before it could be too late. In this approach we consider soft-deadlines as a dangerous but not critical battery charge level which affect robot's fitness. Hard-deadlines are considered as a possible (because of partial knowledge) point where, if the robot does not recharge his battery, an unrecoverable final freezing state is possible. Soft vs hard-deadlines force a change in the robot's operation from behavior-based to mission-oriented (hybrid), which guides the robot using the shortest known path to a nearest previously found charging zone.

The rest of the paper is organized as follows. In Section 2 a brief description of our previously defined model is given. In Section 3 the real-time extension of our model is presented. In Section 4 we show the experimental setup and test results. Finally, in Section 5 some conclusions and future work are drawn.

2 Soft-computing in Robotic Behavioral Control

Much recent progress in robotic navigation has relied on soft-computing (e.g. Fuzzy logic) based methods for their sucess [8–10]. Fuzzy logic has been a mainstain of several efforts in this direction: using independent distributed fuzzy agents and weighted vector summation via fuzzy multiplexers for producing drive and steer commands [10], neuro-fuzzy controllers for behavior design [11], fuzzy modular motion planning [12], fuzzy integration of groups of behaviors [13], multiple fuzzy agents for behavior fusion [14], GA based neuro fuzzy reinforcement learning agents [15], and fuzzy logic integration for robotic navigation in challenging terrain [16]. Our research shows that fuzzy logic has not seen wide usage in robotics in terms of motivating actions and behaviors. We have implemented such motivations (e.g. a need, desire or want) as fuzzy fitness functions for robotic behaviors that serve to influence the intensity and direction of robotic behaviors. Motivation is generally accepted as involved in the performance of

learned behaviors. That is a learned behavior may not occur unless it's driven by a motivation [17]. Differences in motivations help to produce a variety of behaviors which have a high degree of benefit (or fitness) for the organism.

In our experiments, we used motivation settings to determine the fuzzy fitness of a robot in various environments. In terms of robotic learning the motivations that we consider include: curiosity (C), homing (H), and energy (E), the opposite of laziness). There are five triangular membership functions used for each of the four motivations in our experiment (Very Low, Low, Medium, High, Very High).

Takagi-Sugeno-Kang (TSK) fuzzy logic model is used, TSK fuzzy logic does not require deffuzification as each rule has a crisp output that is aggregated as a weighted average [18]. In our method (Fig. 1), the fuzzy motivations considered include the parameters of C, H, and E, which are used as input settings (between 0 and 1) prior to running each experiment. A run environment (room) is selected and the GA initial robot population is randomly initialized. After this, each robot in the population performs its task (navigation and optionally environment recognition) and a set of fitness values (a, g, b) corresponding to the performed task are obtained.

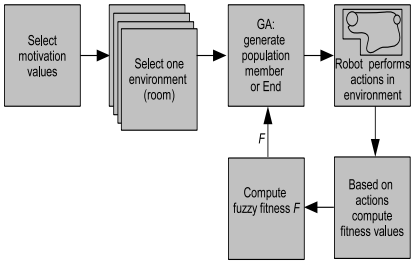


Fig. 1. System Overview

The fitness criteria and the variables that correspond to them are: amount of area explored (a), proper action termination and escape from original neighborhood area (g), and percent of battery usage (b). These fitness values are calculated after the robot completes each run. The a value is determined by considering the percentage area explored relative to the optimum, g is determined by $g = 1 - \frac{l}{L}$, where l is the final distance to robot's home and L the theoretical maximum value. Finally b is the estimated total energy consumption of the robot considering each step.

The final fuzzy motivation fitness value (F) is calculated using TSK based fuzzy logic (three fuzzy variables with five membership functions each: $3^5 = 243$ different fuzzy rules) as shown in Fig. 2 and using the membership functions to compute μ values. For the coefficient array C we used a linear function.

Algorithm FuzzyFitness
Input:
N : number of fuzzy motivations;
M : number of membership functions per motivation;
$X[N]$: array of motivation values preset;
$Y[N]$: array of fitness values;
$C[N]$: array of coefficients;
$\mu[N][M]$: matrix of membership values for each motivation;
Variables:
$w[n]$: the weight for each fuzzy rule being evaluated;
$f[n]$: the estimated fitness;
$n, x_0, x_1, \ldots, x_N$: integers;
Output:
F : the fuzzy fitness value calculated;
begin
$n := 1$;
for each $x_1, x_2, \ldots, x_N := 1$ **step** 1 **until** M **do**
begin
$w[n] := \min\{\mu[1][x_1], \mu[2][x_2], \ldots, \mu[N][x_N]\}$;
$f[n] := \sum_{i=1}^{N} X[i]Y[i]C[x_i]$;
$n := n + 1$;
end;
$F := (\sum_{i=1}^{N^M} w[i]f[i])/(\sum_{i=1}^{N^M} w[i])$;
end;

Fig. 2. Fuzzy Fitness Algorithm

2.1 Implementation

The YAKS (Yet Another Khepera Simulator) simulator is the base for our implementation. YAKS is a simple open source behavior-based simulator [7] that uses neural networks and genetic algorithms to provide a navigation environment for a Khepera robot. Sensors are directly provided into a multilayer neural network in order to drive left and right wheel motors. A simple genetic algorithm is used with 200 members, 100 generations, mutation of 1%, and elite reproduction. Random noise (5%) is injected into sensors to improve realism. The GA provides with a mechanism for updating neural network weights used by each robot in the population that is being optimized. An overview of our fuzzy fitness implementation is shown in Fig. 3.

Outputs of the Neural Network are real valued motor commands (Motor_L and Motor_R) between 0 and 1 which are discretized into one of four actions (left 30°, right 30°, turn 180°, go straight). This follows the Action-based environmental modeling (AEM) search space reduction paradigm [19].

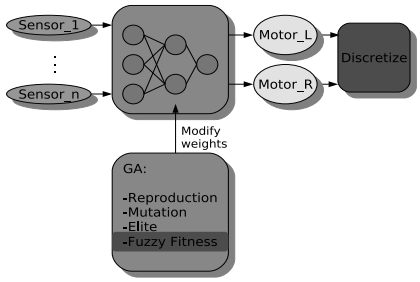


Fig. 3. Fuzzy fitness implementation

3 Real-Time Extensions

During environment exploration, autonomous or semi-autonomous mobile robots are confronted with events which could be predictable such as walls and static objects, or unpredictable such as moving objects or environmental changes. Some of these events must be attended in real-time (responsiveness) to guarantee the robot's integrity (survivability) [20].

Traditional control mechanisms are based on reliable real-time systems, i.e. time constraints over executions and predictability [21], also known as dependable systems [22], e.g. the mars pathfinder or DUSAUV, a semi-autonomous underwater vehicle presented in [23]. On the other hand, soft-computing based methods have not been widely used in this arena due to their inherent uncertainty.

In order to introduce real-time considerations into our behavior-based mobile robot for a real-world application, we extend our model by using temporal constraints during the navigation test-phase. The constraints considered include energy consumption and finite battery charge capacity.

In our approach, we define soft-deadlines as a dangerous but not critical battery charge level which dynamically affects robots behavior. This could influence behaviors to avoid highly energy consuming actions and guiding the robot's movement to some recharging zone if necessary.

When a critical battery level is reached, the previously defined method is no longer useful. A responsive real-time method is needed to, if possible, guarantee survivability [20]. Strictly speaking, we can't guarantee survivability (also beyond the scope of this paper) because of the robots partial knowledge of the world map which, initially, has no recharge zones mapped (we do not consider the starting point as a recharging zone). Nevertheless, because of the off-line robot training-phase, we expect that the trained robot (e.g. NN) will be capable of finding charging zones during the testing phase. Using the charge zone information obtained on-line, the robot applies real-time navigation. We establish a hard-deadline as the point of the robot's unrecoverable final freezing state. Before reaching this deadline (with a 10% safety margin) the robot's operation

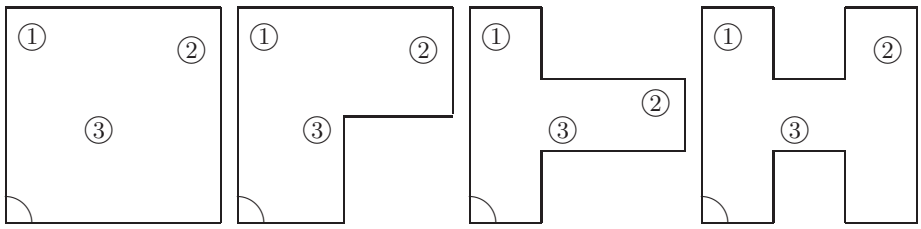
mode changes from behavior-based to mission oriented, following the shortest path to the nearest previously found charging zone [24].

In this paper we focused our research on robots survivability and exploration capability analysis.

4 Experimental Evaluation

The major purpose of the experiments reported in this section is to study the influence of our real-time extensions over the robot's behavior, considering survivability and exploration capability.

We have designed four different rooms (environments) for the robot to navigate in. We denote these rooms as: S-ROOM (the simplest), L-ROOM, T-ROOM and H-ROOM (most complex). Walls are represented by lines and we designate up to three charging zones (see circles in Fig. 4). The starting zones for each room will be the lower left corner (quarter circles in Fig. 4).



(a) S-room (b) L-room (c) T-room (d) H-room

Fig. 4. Experiment rooms layout with starting and recharging zones.

We will denote by NRT the behavior-based algorithm which would operate the robot without any real-time considerations, i.e. the battery level has no influence over robot's behavior but, if it comes near to a charging zone the battery level is updated to his maximum capacity. The main characteristics of NRT are:

- the battery level has no influence on the robot during training phase and,
- there is no input neuron connected to the battery sensor.

We denote by SRT the algorithm which would operate the robot with soft-real time considerations, influencing his behavior to avoid a dangerous battery level. This algorithm differs from NRT mainly by

- battery level influences robot's fitness evaluation used by the GA and,
- a new input neuron is connected to a battery level sensor.

Finally, we denote by HRT the hybrid algorithm which would operate the robot with hard-real time considerations, i.e., the same as SRT incorporating critical battery level sensing, and also having the capacity to change the robot's normal operation to mission oriented, guaranteeing his survivability (if at least one charging zone was previously found).

4.1 Experimental Setup

As mentioned before, the experiments are performed using a modified version of YAKS [7]. This simulation system has several different elements including: the robot simulator, neural networks, GA, and fuzzy logic based fitness.

Khepera Robot For these simulations, a Khepera robot was chosen. The robot configuration has two DC motors and eight (six front and two back) infrared proximity sensors used to detect nearby obstacles. These sensors provide 10 bit output values (with 5% random noise), which allow the robot to know in approximate form the distance to local obstacles. The YAKS simulator provides the readings for the robot sensors according to the robot position and the map (room) it is in. The simulator also has information for the different areas that the robot visits and the various obstacles (walls) or zones (home, charging zones) detected in the room. In order to navigate, the robot executes up to 1000 steps in each simulation, but not every step produces forward motion as some only rotate the robot. If the robot has no more energy, it freezes and the simulation stops.

Artificial Neural Network The original neural network (NN) used has eight input neurons connected to the infrared sensor, five neurons in the hidden layer and two output neurons directly connected to the motors that produce the robot movement. Additionaly, in our real-time extensions we introduce another input neuron connected to the battery sensor (activated by Srt and Hrt).

Genetic Algorithm A GA is used to find an optimal configuration of weights for the neural network. Each individual in the GA represents a NN which is evolving with the passing of different generations. The GA uses the following parameters:

- Population size: 200
- Crossover operator: random crossover
- Selection method: elite strategy selection
- Mutation rate: 1%
- Generations: 100

For each room (see Fig. 4) we trained a robot up to 400 steps, considering only configurations with 2 or 3 charging zones, i.e. shutting down zone 3 for 2-zones simulations. Startup battery level allows the robot to finish this training phase without recharging requirements.

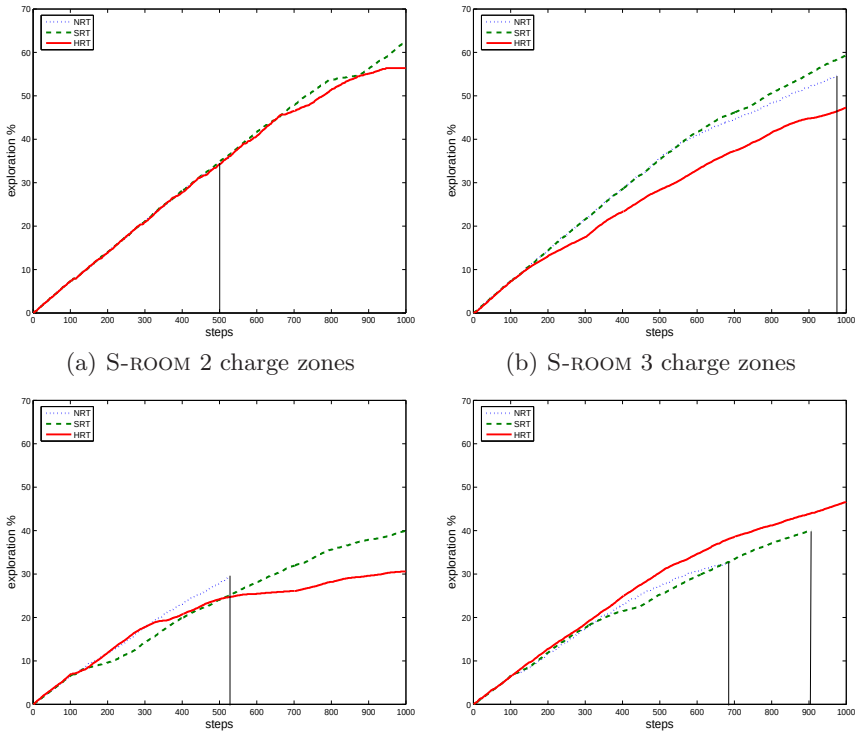
Finally, we tested our algorithms in each room up to 1000 steps, using the previously trained NN for each respective room. The startup battery level was set to 80 (less than 50% of it's capacity), which was insufficient to realize the whole test without recharging.

4.2 Experimental Results

Due to size restrictions, we selected representative behaviors for only 2 rooms. We chose the S-ROOM and H-ROOM to show results for a simple and complex room respectively.

In Fig. 5 we show the robot's exploration behavior for selected rooms. Each curve in the graph shows the average value of 10 executions of the same experiment (deviation between experiment iterations was very small, justifying only 10 executions). Let $surv(\alpha)_i$ the survivability of the experiment instance i of algorithm α, we define $surv(\alpha)$ as the survivability of an experiment applying algorithm α as the worst case survivability instance of an experiment, i.e.

$$surv(\alpha) = \min_{i=1,\ldots,10} [surv(\alpha)_i] \tag{1}$$



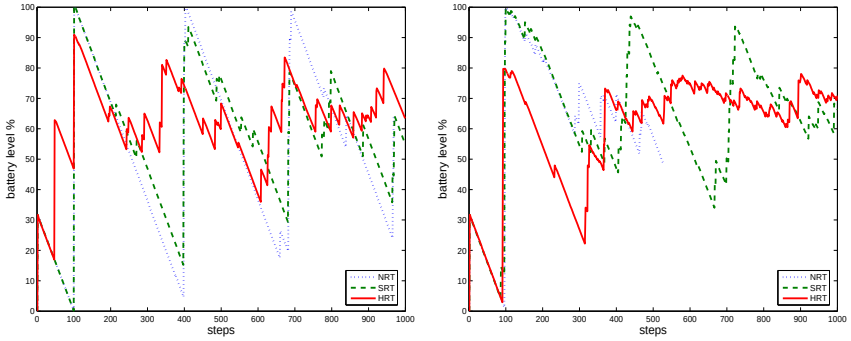
(a) S-ROOM 2 charge zones (b) S-ROOM 3 charge zones

(c) H-ROOM 2 charge zones (d) H-ROOM 3 charge zones

Fig. 5. Exploration Behavior

Please note that the end of each curve in Fig. 5 denotes the survivability of the respective algorithm (for better readability, we mark NRT survivability with

a vertical line). Reaching step 1000 (the maximum duration of our experiments) means the robot using the algorithm survives the navigation experiment. Finally, in Fig. 6 we show a representative robot's battery level. Monitoring was made during test phase in a H-ROOM with 3 charging zones.



(a) S-ROOM 3 charge zones (b) H-ROOM 2 charge zones

Fig. 6. Battery Behavior

4.3 Discussion

The results of our experiments are summarized below:

Survivability: As shown in Fig. 5, SRT and HRT algorithms give better reliability of completing missions than the NRT method, independently of the rooms (environments) we use for testing (see Fig. 4). As expected, if fewer charging zones are provided, NRT has a less reliable conduct. Please note that as shown in Fig. 6, NRT is also prone to battery depletion risk and does not survive in any case.

When varying room complexity, i.e. 5(b) and 5(d), real-time considerations have significant impact. Using SRT, a purely behavior-based driven robot (with the additional neuron and motivation), improves it's performance.The SRT method does not guarantee survivability since without using real-time the robot is prone to dying even with greater number of recharge zones (as seen in 5(d)). Finally, we conclude that despite the uncertainty introduced by soft-computing methods, HRT (e.g. the hybrid algorithm), in general is the best and safest robot control method from a real-time point of view.

Exploration Environment: As can be seen in Fig. 5 safer behaviors mean slower exploration rates (more conservative), up to 12% slower in our experiments. When comparing NRT with SRT, the exploration rates are almost equal

in simple environments. In more complex rooms, SRT exploration is slower than NRT (due to battery observance). However, because of SRT having better survivability on the whole it's performance wins over NRT. If we compare NRT with HRT, exploration performance also favors NRT, wich could be explained given HRT conservative battery management (see Fig. 6).

Given 2 charge zones, HRT behaves differently in environments of varying complexity (up to 25%) which could be attributed to the complexity of the returning path to the nearest charging zone and loosing steps in further exploration. This phenomena becomes less notoriosly when increasing the number of charging zones (more options for recharge).

5 Conclusions and Future Work

In this paper we investigate real-time adaptive extensions of our fuzzy logic based approach for providing biologically based motivations to be used in evolutionary mobile robot learning. We introduce active battery level sensors and recharge zones to improve robot's survivability in environment exploration. In order to achieve this goal, we propose an improvement of our previously defined model (e.g. SRT), as well as a hybrid controller for a mobile robot (e.g. HRT), combining behavior-based and mission-oriented control mechanisms.

These methods are implemented and tested in action sequence based environment exploration tasks in a Khepera mobile robot simulator. Experimental results shows that the hybrid method is in general, the best/safest robot control method from a real-time point of view. Also, our preliminary results shows a significant improvement on robot's survivability by having minor changes in the robot's motivations and NN. Currently we are designing a real robot for environment exploration to validate our model moving from simulation to experimentation. Improving dependability of HRT, we want to extend this control algorithm to safety-critical domains.

References

1. Jeen-Shing Wang and C.S. George Lee, Self-adaptive recurrent neuro-fuzzy control of an autonomous underwater vehicle, IEEE Transactions on Robotics and Automation, Volume 19, Issue 2, Apr 2003, Pages 283-295.
2. Rahul Kumar Jha, Balvinder Singh and Dilip Kumar Pratihar, On-line stable gait generation of a two-legged robot using a genetic-fuzzy system, Robotics and Autonomous Systems, Volume 53, Issue 1, , 31 October 2005, Pages 15-35.
3. Seraji, H.; Howard, A., Behavior-based robot navigation on challenging terrain: A fuzzylogic approach, IEEE Transactions on Robotics and Automation, Volume 18, Issue 3, Jun 2002, Pages 308 - 321.
4. Guido Maione and David Naso, A soft computing approach for task contracting in multi-agent manufacturing control, Computers in Industry, Volume 52, Issue 3, Soft Computing in Industrial Applications, Dec. 2003, Pages 199-219.
5. Tom Ziemke, Dan-Anders Jirenhed and Germund Hesslow, Internal simulation of perception: a minimal neuro-robotic model, Neurocomputing, Volume 68, , October 2005, Pages 85-104.

6. Bernhard Sick, Markus Keidl, Markus Ramsauer, and Stefan Seltzsam A Comparison of Traditional and Soft-Computing Methods in a Real-Time Control Application ICANN 98, Proc. of the 8th Int. Conference on Artificial Neural Networks; pages 725-730, Sweden, Sep. 1998.
7. YAKS simulator website: http://r2d2.ida.his.se/
8. Jang, J., Chuen-Tsai, S., Mitzutani, E.: Neuro-Fuzzy and Soft Computing, Prentice Hall, NJ, (1997).
9. Konolige, K., Meyers, K., Saffiotti, A.: FLAKEY, an Autonomous Mobile Robot. SRI technical document, July 20 (1992)
10. Goodrige, S., Kay, M., Luo, R.: Multi-Layered Fuzzy Behavior Fusion for Reactive Control of an Autonomous Mobile Robot. Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (July 1997) 573-578
11. Hoffman, F.: Soft computing techniques for the design of mobile robot behaviors. Information Sciences, 122 (2000) 241-258
12. Al-Khatib, M., Saade, J.: An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot. Fuzzy Sets and Systems, 134 (2003) 65-82
13. Izumi, K., Watanabe, K.: Fuzzy behavior-based control trained by module learning to acquire the adaptive behaviors of mobile robots. Mathematics and Computers in Simulation, 51 (2000) 233-243.
14. Martnez Barber, H., Gmez Skarmeta, A.: A Framework for Defining and Learning Fuzzy Behaviours for Autonomous Mobile Robots, International Journal of Intelligent Systems, 17(1) , (2002) 1-20
15. Zhou, C. : Robot learning with GA-based fuzzy reinforcement learning agents, Information Sciences, 145 (2002) 45-68
16. Seraji, H., Howard, A.: Behavior-Based Robot Navigation on Challenging Terrain: A Fuzzy Logic Approach, IEEE Trans on Robotics and Automation, V.18, No.3 (June 2002) 308-321
17. Huitt, W.: Motivation to learn: An overview. Educational Psychology Interactive. Valdosta State University.: http://chiron.valdosta.edu/whuitt/col/motivation/-motivate.html, (2001).
18. Jang, J.-S, Sun, C.-T., Sun, Mizutani, E.: Neuro-Fuzzy and Soft Computing: a computational approach to learning and machine intelligence, Prentice Hall, NJ, (1997).
19. Yamada, S.: Evolutionary behavior learning for action-based environment modeling by a mobile robot. Applied Soft Computing, 5 (2005) 245-257.
20. Hermann Kopetz. Real-Time Systems Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, Boston, Massachusetts, april 1997.
21. Gheith, A. and Schwan, K. 1993. CHAOSarc: kernel support for multiweight objects, invocations, and atomicity in real-time multiprocessor applications. ACM Trans. Comput. Syst. 11, 1 (Feb. 1993), 33-72.
22. G. Motet and J. -C. Geffroy, Dependable computing: an overview, Theoretical Computer Science, Volume 290, Issue 2, 2 January 2003, Pages 1115-1126.
23. Ji-Hong Li, Bong-Huan Jun, Pan-Mook Lee and Seok-Won Hong. A hierarchical real-time control architecture for a semi-autonomous underwater vehicle. Ocean Engineering, Volume 32, Issue 13 , September 2005, Pages 1631-1641.
24. P. Tompkins, A. Stentz, and D. Wettergreen, Mission-level path planning and re-planning for rover exploration, Robotics and Autonomous Systems, Intelligent Autonomous Systems, Vol. 54, No. 2, February, 2006, pp. 174 - 183.