INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Advanced Robotic Systems

Fuzzy Motivations in a Multiple Agent Behaviour-Based Architecture

Regular Paper

Tomás V. Arredondo[1,*], Wolfgang Freund[1], Nicolás Navarro-Guerrero[2] and Patricio Castillo[1]

1 Departamento de Electronica, Universidad Técnica Federico Santa María, Valparaiso, Chile
2 University of Hamburg, Department of Computer Science, Hamburg, Germany
* Corresponding author E-mail: tomas.arredondo@usm.cl

Received 1 Jun 2012; Accepted 19 Apr 2013

DOI: 10.5772/56578

Abstract In this article we introduce a blackboard-based multiple agent system framework that considers biologically-based motivations as a means to develop a user friendly interface. The framework includes a population-based heuristic as well as a fuzzy logic-based inference system used toward scoring system behaviours. The heuristic provides an optimization environment and the fuzzy scoring mechanism is used to give a fitness score to possible system outputs (i.e. solutions). This framework results in the generation of complex behaviours which respond to previously specified motivations. Our multiple agent blackboard and motivation-based framework is validated in a low cost mobile robot specifically built for this task. The robot was used in several navigation experiments and the motivation profile that was considered included "curiosity", "homing", "energy" and "missions". Our results show that this motivation-based approach permits a low cost multiple agent-based autonomous mobile robot to acquire a diverse set of fit behaviours that respond well to user and performance expectations. These results also validate our multiple agent framework as an incremental, flexible and practical method for the development of robust multiple agent systems.

Keywords Mobile Robot, Multi-Agent, Fuzzy Logic, Motivations, Evolutionary, Blackboard

1. Introduction

Psychologists view motivations as reasons that individuals hold for starting and performing voluntary behaviour. They may also affect an individual's perception, cognition and emotion. Even though they are associated with cognition, an individual may not be aware of having the motives that initiated a behaviour [1]. In general, a learned behaviour may not be executed unless it is driven by a motivation to do so. There are many sources of motivations including: behavioural, social, biological, cognitive, affective and spiritual [2]. Given their role in initiating behaviour, differences in motivations are key drivers in helping to produce a diversity of behaviours with a high degree of benefit (or fitness) for certain organisms.

Developing simple, intuitive and user friendly interfaces between people and complex semi-autonomous artificial systems is clearly a desirable and beneficial objective. Considering this, interfaces need to be developed which are simple for average users to understand and that do not require technical mastery from a typical user. There are many important applications of such technology, for example, in tasks such as industrial inspection, disaster recovery, hazardous environment exploration, web crawling and data mining to name a few. In many of these applications it is not practical or even desirable for the user to directly control robot behaviour and a degree of autonomy is desirable. It is also highly desirable for such systems to be implementable in a relatively low cost manner. Research in these areas is ongoing and there are many examples of these ideas being applied in a variety of automated applications and systems [3–8].

With these objectives in mind, we introduce a blackboard and Mamdani fuzzy logic-based multiple agent system architecture which is tested in a low cost mobile robotic context. In our current scheme, a Mamdani fuzzy logic inference system (FIS) is used to obtain fitness scores for a heuristically produced population of different behaviours. Toward this goal, we select and implement a genetic algorithm (GA)-based agent as part of a long range planning process in order to determine highly fit routes for the robot. A fuzzy logic-based collision avoidance agent is also introduced in this context as a means by which to generate reactive behaviour options as an alternative to the long range planning agent. A monitor agent determines the best behavioural option given current environmental threat conditions and available options. In addition, a novel motivation fulfilment product (MFP) is used as a means to reduce the real computation required by the FIS.

We have successfully evaluated our architecture in various real-world navigation tasks, such as obstacle avoidance, mapping and mission completion, within a real low cost robot specifically designed for this task. The performance implications of this approach versus single agent implementations are also considered.

This paper is organized as follows. Section 2 describes prominent agent architectures. Section 3 presents an overview of fuzzy logic applied to agent behavioural control. Section 4 describes our system design. In Section 5, we summarize the experiments that were conducted. Finally, in Section 6 conclusions are drawn.

2. Agent Architectures

We discuss several relevant agent architectures which we categorize as follows: deductive, deliberative toward means-ends, deliberative motivated, distributed deliberative motivated and behaviour-based approaches.

2.1 Deductive Architectures

These architectures require a symbolic representation of the environment. Deductive agents manipulate these symbols using logical operations in a manner that corresponds to logical deduction. A set of deduction rules is applied when certain conditions are satisfied. There have been many successful systems implemented with deduction rules, but still there are issues that have been identified with these techniques [3, 7, 9–11]:

- Transduction and representation - having a limited number of symbolic values causes an intrinsic inaccuracy in representing the real world with symbols. This issue is exacerbated by having agents manipulate symbolic data while obeying time constraints.
- Reasoning - the knowledge required for reasoning is based on the premise that it be consistent, reliable and certain, but this is not always possible.
- Hierarchical - communication and control flows up and down the agent structure in a sequential manner. Any environmental changes must go completely up the hierarchy in order for the agent to deliberate on a new course of action to be taken. Any decisions must then go down the entire hierarchy for the action to be executed, and this architecture does not typically consider parallel solutions.
- Non incremental - implementation generally requires building the entire system before system testing can be done. This is contrary to current methods involving agile, iterative or incremental engineering processes [12, 13].
- Changing environments - required corrective actions due to environmental changes lead to a large reaction latency.

2.2 Deliberative Toward Means-Ends Architectures

Deliberation toward means-ends reasoning is an action directed architecture where the basic system objective is to determine what to do next. The belief desire intention (BDI) paradigm follows this approach with explicitly represented data structures that correspond to these three mental states [3].

The procedural reasoning system (PRS) was an implementation of this approach that was applied in a variety of agent-based applications [14–16]. PRS consists of: (i) a database (i.e. beliefs) that has facts about the environment and internal states, (ii) desires representing agent’s goals, (iii) an intention structure with the plans the agent is committed to achieve, and (iv) a set of plans that describe how the action sequences with satisfied preconditions are to be executed to fulfil certain desires. PRS plans include the following:

- a goal - the plan postcondition,
- a context - the plan precondition, and
- a body - the plan recipe with the set of actions to execute.

The entire system is managed serially by an interpreter that selects the appropriate plans determined by current beliefs and desires. The selected plans are placed on the intention structure so that they may be executed.

A deliberative agent generates goals in-line given a changing domain. Unfortunately, there are resource constraints when multiple goals have been specified. Determining whether a propositional planning instance has solutions is a PSPACE-complete problem. This may be reduced to an NP-complete problem provided restrictions are placed on the operators and the formulas used, hence, there is no known efficient way to locate a solution [11, 17].

In order to reduce the amount of reasoning that a BDI agent has to do, a filtering strategy can be applied. In the approach of Bratman et al. [14], options are filtered on the basis of similarity to current intentions. As previously reported, these mechanisms may not be very successful in reducing cognitive overload because of the multiplicity of valid alternatives over time [11, 18].

2.3 Deliberative Motivated Agents

In deliberative schemes, motivations have been used to represent the basic needs of agents and to model the environmental context. In general, deliberative motivated agents associate with each motivation a numeric measure of strength or motivational value (i.e. *intensity*).

The context of an agent is determined by the agent's sensors, actuators, size, internal states and so on. An important part of this context is obtained when reasoning using current motivations. These motivations constrain the *desires* that the agent can generate and enable the agent to prioritize its goals and perform plan selection [19].

In this architecture, goals are activated depending on their importance, temporal relevance and the current available cognitive resources. In generating a motivation value, a balance must be achieved between the subjective importance versus temporal relevance. If this balance is not found, the agent will tend to behave in undesirable ways: it will only consider the most important goals without temporal considerations or if the temporal factor of the goal has too much of an influence on motivation intensity, it will only consider its most urgent goals without considering their importance [11, 18].

2.4 Distributed Deliberative Motivated Agents

Davis et al. [10] propose a distributed BDI-based model of motivations. In this work, motivation related values (e.g. belief indicator, commitment, dynamic state, intensity, urgency, decay) are continuously updated with an associated magnitude in the range [0,1] that can fluctuate according to success or failure as mapped by other processes. In this approach, the robo-CAMAL system controls the actions of a mobile robot through the use of a BDI schema, a motivational blackboard, motivational control states and fixed reactive behaviours. robo-CAMAL has the ability to anchor events and objects to pre-defined symbols. It uses a domain model to recognize objects (e.g. a *blueball*) and events (e.g. a *hit* (*blueball*)).

Associations are made that consist of a belief set (*BelSet*), a desire, an intention and an association value (*Insistence*). The associations provide an indication of the past success of a specific set of plans given an agent's current beliefs and desires. In this fashion the agent can consistently determine the most appropriate set of plans based on its beliefs and desires. Associations take the following form: *association* (*BelSet*, *Goal*, *BehaviourSpecs*, *Insistence*).

robo-CAMAL can build new associations based on initialization or through a goal failure mechanism. This mechanism can generate all possible belief-goal-intention combinations where each is given a default association value. robo-CAMAL can adapt these associations through the modification of the affect value. Here, a learning algorithm based on reinforcement learning with an immediate reward is utilized. Reinforcement learning involves an agent performing an action within its environment after which it receives a reward or penalty based on the result of that action. By trying different actions and using this feedback, the agent learns how to maximize the total reward. robo-CAMAL learns by using a simplified Q-learning approach. The association value can be written as: $A(s; g; a)$, where s is the state, g is the goal and a is the action [10].

robo-CAMAL has demonstrated an ability to anchor symbols to perceptual data with the use of a domain model. This system performs very well in a controlled environment, but has difficulty in more unstructured environments. This is apparently due to a lack of sophistication in the domain and learning models [10]. As previously indicated in Section 2.2, there are other well known learning issues (e.g. cognitive overload) in BDI-based architectures which could also be partially responsible for these problems by limiting the depth of available processing.

2.5 Behaviour-Based Architectures

Behaviour-based agent architectures in general try to avoid internal model representations and symbolic information because of their inherent error. Hence, the main advantage in behaviour-based methods lies in the tight coupling between sensing and acting. Using this technique, agents can be constructed in an inexpensive and incremental manner. In addition, they can be uncalibrated, robust to noise and can behave well - even with simple low cost computers and communication infrastructure [7, 9, 11].

In general, a purely reactive agent $(\mathrm{Ag_r})$ produces an individual response out of the set of all possible responses (R) based on a current stimulus (S):

$$\mathrm{Ag_r} : \beta(\mathrm{S}) \rightarrow \mathrm{R} \quad (1)$$

2.5.1 Behaviour Function

β_i (where $\beta_i \in \beta$) is the behaviour function responsible for mapping stimuli into responses. This function can be discrete or continuous [7]:

- Discrete: the response to stimulus is one out of a finite set of choices (e.g. go-straight, turn-right, turn-left). Where R is the bounded set of responses to stimuli S as specified by β. There is no general restriction as to the implementation of the discrete behaviour function.
- Continuous: the response to stimulus is mapped over a continuous range R. This allows an unbounded space of possible responses (e.g. potential field method) [7].

2.5.2 Behaviour Layering and Coordination

As previously mentioned, behaviour-based systems are developed in an incremental manner. Different behaviour functions are implemented on top of one another iteratively. Contrary to the vertical layering pursued by the AI community, this approach is structured horizontally. [11].

In horizontal architectures, parallelism is an inherent feature, but higher layers may still manage lower layer outputs by inhibiting or suppressing them (e.g. subsumption) [7, 9]. Parallel layers can act independently, but this approach generally requires a coordinator to ensure consistency of behaviour and to determine which layer determines overall system output.

Coordination mechanisms may be competitive or cooperative. In competitive methods the coordinator typically will select a winner from many competing options. This method of coordination can be done in several ways: by strictly suppressing lower layers (e.g. subsumption) [9], by arbitrarily selecting the output of a single layer based on activation levels as determined by goals and current stimulus, and using voting-based methods. In cooperative methods, all offered behaviours are fused or somehow integrated. This may be done using vector summation with individual behaviour gains [7].

2.5.3 Motivations in Behaviour-Based Mobile Robot Systems

Nourbakhsh et al. [20] implemented a museum guide robot using a fuzzy state model, reactive case-based navigation and robotic moods. This robot used five different moods to escape from a trapped condition by communicating to museum visitors (i.e. obstacles) that it was "frustrated".

Michaud and Vu [21] extended the behaviour-based motivation approach by implementing an architecture where subsumption-based behaviour modules and internal motivations were used by a *final selection* module that activated or deactivated behaviours using a Boolean function given their desirability and activation thresholds. They explored using specific visual signals as dictionary-based queues to a *cognition* module. Each encoded signal determined discrete excitatory or inhibitory changes to energy levels of motives toward obtaining certain goals (e.g. wandering, foraging, wall following, following).

An architecture using synthetic emotions was developed by Breazeal [22] in order to influence the internal dynamics of a social robot's controller by biasing attention, motivation, motor behaviours and learning. The robot obtained responses to its actions from a human attendant by recognizing visual queues. The motivation system consisted of drives and emotions. Drives represented the basic needs of the robot including: a social interaction drive, a need to be stimulated with toys and other objects, and a need for rest (i.e. fatigue drive). For each drive, there was a desired operation point and bounds of operation (i.e. a homeostatic regime). This line of research has been extended by several other authors [23].

Stoytchev and Arkin [24] investigated motivations (i.e. curiosity, frustration, homesickness and anger) in the context of internal state variables used by a fax delivery robot to manage its high level strategy (e.g. when to work for its mission, when to look for help, when to head for home, etc.).

Implementing moral and ethical aspects of autonomous mobile robot (AMR) behaviour has been investigated by [25]. In his proposed approach, Bayesian networks model human and environmental health, a decision network manages utility values and a Markov decision process decomposes the planning problem which needs solving in order to obtain an optimal course of action.

3. Fuzzy Logic in Agent Behavioural Control

There is a large body of research in applications of fuzzy logic in the context of intelligent systems [26–29]. In general, fuzzy logic-based systems make decisions using fuzzy rules and variables. Fuzzy variables in the rules indicate the degree of membership of the variable to a fuzzy set. This degree of membership (between 0 and 1) is defined by a function known as the membership function, which maps a crisp input to a fuzzy output (fuzzifier). Fuzzy logic systems consist of a fuzzifier, a fuzzy rule engine and a defuzzifier which converts fuzzy values back to crisp values.

In a manner consistent with behaviour-based systems, fuzzy logic aims to provide a more natural interface with a larger tolerance for imprecision than traditional logic and which may be optimized to provide useful results in a variety of applications [26, 29]. Many behaviour-based agent applications have used fuzzy logic. Flakey and Marge [30, 31] are two robots that used fuzzy-based implementations to coordinate different behaviours such as collision avoidance, goal seeking, docking and wall following. One issue with fuzzy rule-based systems is that they suffer from having an exponential growth in the fuzzy rule base. This issue with scalability may be managed in a variety of ways: by a reduction of input spaces, by using limited world model contexts [30], or with independent fuzzy agents weighted by vector summation via fuzzy multiplexers [31].

Some fuzzy logic strategies that have been used in mobile robotics include: fuzzy-based modular motion planning [32], neuro-fuzzy controllers for behaviour design (based on direction, distance and steering) [33], genetic algorithm-based neuro fuzzy reinforcement learning agents applied toward training a walking robot [34], fuzzy integration of groups of behaviours [35], multiple fuzzy agents used in behaviour fusion [36], behaviour-based fuzzy logic integration for robotic navigation in challenging terrain [37], adaptive fuzzy behaviour hierarchies termed multiple composite levels (MCL) in a hybrid learning/reactive multiple agent control system applied in a simulated flocking application [38], and motivations as Takagi-Sugeno-Kang fuzzy fitness functions applied in a simulated single robot navigation experiment using a neural network as the behaviour generator agent [8, 11].

4. Fuzzy Motivations for Multiple Agent Learning

As previously mentioned, motivations are essential for producing different behaviours which derive in fitter organisms. In our proposed multiple agent approach, motivations are not used to reduce the search space of behaviours as is the case with deliberative motivated multiple agent implementations, but to provide a scoring mechanism for agent behaviour. This method also does not use a separate goal activation system with associated triggers and guidelines given their known associated computational liabilities.

Our multiple agent method allows the user to manage system behaviour indirectly while still maintaining agent autonomy. User expectations of system behaviour are mapped via a fuzzy motivation profile that directly influences agent behaviour. Other motivation-based methods focus only on the agent and the behaviours required for it to complete tasks but without end user consideration or do so very indirectly through an additional emotional layer.

4.1 Agent Definition

In our system, S is the domain of robot perceived stimuli, M is the fuzzy motivation set previously described, β is the behaviour function and R is the obtained responses. The motivated behaviour-based agent Ag_{mbb} can be defined as:

$$Ag_{mbb} : \beta(S,M) \rightarrow R \qquad (2)$$

M is fixed as initially set by the end user throughout the experiment. This provides for a constant influence on robot behaviour.

4.2 Fuzzy Fitness

In our current implementation, we apply a Mamdani fuzzy logic model [27] in order to implement a fuzzy inference system (FIS). This FIS, calculates fuzzy fitness scores which are used to select the fittest agent from available candidates. We have used triangular fuzzy functions toward determining agent fitness as shown in Fig. 1.

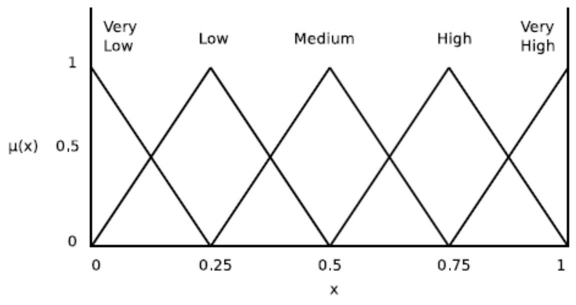
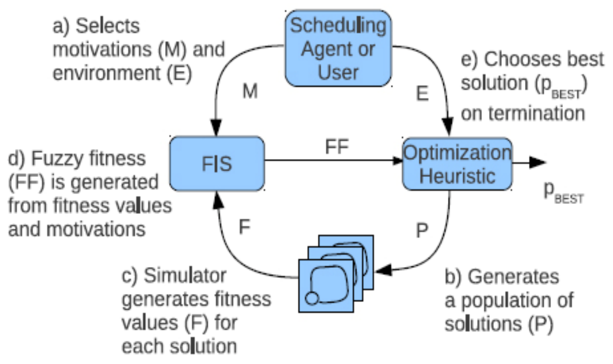


Figure 1. Fuzzy membership functions

Let $E = \{e_1, e_2, \dots, e_L\}$ be a set of (L) simulated environments available for the agent to run in. Also let $M = \{m_1, m_2, \dots, m_K\}$ be a fuzzy motivation set (M) consisting of K different motivations. Fuzzy motivations are used as normalized fixed input settings prior to having an agent perform a run in the environment. As shown in Fig. 2, in (a) the user or agent selects a motivation profile (M) and an environment (E). Then, a population of N solutions $P = \{p_1, p_2, \dots, p_N\}$ (e.g. different runs in the environment) is generated by an optimization heuristic (b). After this, each individual solution in the population performs behaviours in the simulated environment (c) and a set of fitness values (F) corresponding to each performance is obtained (d). A fuzzy fitness (FF) is calculated by the Mamdani FIS for each member of the population using the fitness criteria information previously obtained (F) and the different motivations (M) at the time of exploration. The individual solution (i.e. p_{BEST}) with the best score is selected as the final solution (e).



a) Selects motivations (M) and environment (E)
Scheduling Agent or User
e) Chooses best solution (p_{BEST}) on termination
M
E
d) Fuzzy fitness (FF) is generated from fitness values and motivations
FIS
FF
Optimization Heuristic
p_{BEST}
F
P
c) Simulator generates fitness values (F) for each solution
b) Generates a population of solutions (P)

Figure 2. Fuzzy fitness system overview

4.3 Fuzzy Algorithm

The algorithm used by the Mamdani FIS is shown in Fig. 3. As seen in this algorithm, the number of rules required to compute fuzzy fitness is $\mathrm{R} = \mathrm{N}^{\mathrm{K}}$, where N is the number of fuzzy membership functions used and K is the number of motivations in M. The complexity is $\mathrm{O}\left(\mathrm{G}^{\mathrm{R}}\right)$ since the Mamdani fuzzy fitness (FF) calculation is also dependent on the number of divisions used for centroid calculations (i.e. the granularity G of the discretization used). This system is in general simple to change and enhance with additional motivations and fitness scores. One potential issue is rule bloat which is discussed in the following section.

```
Algorithm FuzzyFitness
Input:
  $K$ : number of fuzzy motivations;
  $N$ : number of membership functions per motivation;
  $R$ : number of fuzzy rules ($R = N^K$);
  $G$ : number of divisions used for centroid calculation;
  $X[1 \ldots K]$ : array of motivation values preset;
  $Y[1 \ldots K]$ : array of fitness values;
  $\mu[R][1 \ldots K]$ : matrix of membership values for each $MFP$;
  $\mu_O[R][G]$ : matrix of output membership functions values;
Variables:
  $\mu_R[R]$ : array of membership values for each rule;
  $\mu_A[G]$ : array of activation values aggregated over all rules;
  $x$ : integer;
Output:
  $F$ : the fuzzy fitness value calculated;
begin
  for $x := 1$ step 1 until $R$ do
    $\mu_R[x] := \min\{\mu[x][1], \mu[x][2], \ldots, \mu[x][K]\}$;
  for $x := 1$ step 1 until $G$ do
    $\mu_A[x] := \max\{\min\{\mu_R[1], \mu_O[1][x]\},$
      $\min\{\mu_R[2], \mu_O[2][x]\}, \ldots, \min\{\mu_R[R], \mu_O[R][x]\}\}$;
  $F := (\sum_{i=1}^{G}((1.0/G) \cdot \mu_A(i))) / \sum_{i=1}^{G} \mu_A(i)$;
end;
```

Figure 3. Fuzzy inference system algorithm

4.3.1 Motivation Fulfilment Product

The motivation fulfilment product (MFP) is a normalized membership value (range from 0 to 1) that helps to reduce the number of rules required by the Mamdani FIS while providing a direct mapping of motivation fulfilment. It is obtained as the product of each motivation with its associated fitness score (e.g. $\mathrm{m}_1\mathrm{f}_1$). These products are used as inputs into the Mamdani fuzzy inference system consisting of K fuzzy variables with N triangular membership functions each (e.g. if $\mathrm{K} = 4$ and $\mathrm{N} = 5$ there are $\mathrm{N}^{\mathrm{K}} = 5^4 = 625$ different fuzzy rules). If the MFP was not used, motivations as well as the associated fitness score would have to be used as fuzzy inputs and the number of rules would be much higher (e.g. $5^8 = 390{,}625$ fuzzy rules).

It is possible that multiple fitness scores could be associated with a particular motivation (e.g. $\mathrm{m}_1\mathrm{f}_1\mathrm{f}_2$), but to enable an evaluation of the feasibility of our method we have chosen the simplest approach of using one fitness score per motivation in the MFP. It is clear that there can be inter-dependencies between motivations (e.g. homing and energy are tightly related) as they are not all behaviourally orthogonal. In our experimental comparisons, we have chosen motivations that are as behaviourally distinct as possible (e.g. curiosity and homing, curiosity and energy, curiosity and missions) to enable simpler analysis.

4.4 Motivation Profile Selection

In the experiments currently considered, the motivations chosen included: Curiosity (m_1), Homing (m_2), Energy (m_3) and Missions (m_4). These were chosen because they correspond to the different levels of the requirements for survival and growth as seen by psychologists [42]. Homing and Energy are part of basic "physiological" needs, "cognitive" include Curiosity as a motivation, and the "self-actualization" need includes task completion (i.e. Missions), "safety" is another need that is considered and is implemented by means of a Monitor process that is constantly evaluating threats to the system (e.g. if an object is too close to the robot) and makes decisions as and when required. From our point of view, safety is the most important layer in this modified hierarchy and contrary to Maslow's hierarchy of needs [42] supersedes physiological needs such as conserving energy, recharging (i.e. food) or waste disposal.

The fitness variables (i.e. F) were chosen so that they would align with the motivations previously described (i.e. M) within the MFP framework. They are: the amount of area explored (f_1), proper action termination and return to original neighbourhood area (f_2), battery usage (f_3) and mission accomplishment score (f_4). These values are normalized and are calculated as follows:

- f_1: percentage area explored relative to the theoretical maximum,
- f_2: 1 - final distance to home/maximum distance to home,
- f_3: estimated percent total energy consumption considering all steps taken,
- f_4: percentage mission accomplishment considering all requested missions (the specific definition of a mission is context dependent and is detailed in the experimental evaluation section).

4.5 Genetic Algorithm

As shown in Fig. 2, our approach requires an optimization heuristic in order to select the best path from a population of possible robot routes in the environment. We selected the genetic algorithm (GA) to perform the required optimization heuristic given its ease of implementation and well documented quality of results in a variety of optimization and path planning tasks [27]. The GA that was implemented follows the traditional means by which to perform route optimization (further details on the implementation are provided in Section 4.8):

- Initially generates a random population of path planning solutions.
- Uses the fuzzy fitness algorithm to compute a fitness score for each solution after path execution in the robot map.
- Scores are used to perform selection for crossover of the chosen solutions using a fitness proportionate scheme.
- Finally, mutation is implemented in the new population obtained in each iteration.

4.6 Blackboard-Based Agent Architecture

Our system uses a multiple agent (i.e. experts) blackboard-based architecture. The idea is to solve a problem by using multiple agents, each agent has its area of expertise and is only responsible for solving that part of the overall problem. An agent can read the results from others using the blackboard and a coordinator agent (i.e. the Monitor) decides priorities if there are any conflicts.

As seen in Fig. 4, the software consists of agent experts that operate autonomously in parallel. The interface between the experts is known as the common data area (CDA) and serves as the required blackboard. In the CDA, each expert has a space where its results are saved (e.g. the mapper agent stores the environment map in the CDA). Synchronization of CDA access is arranged with semaphores to manage mutually exclusive interactions. This means that each process may block, if necessary, on an area in the CDA when reading or writing data.

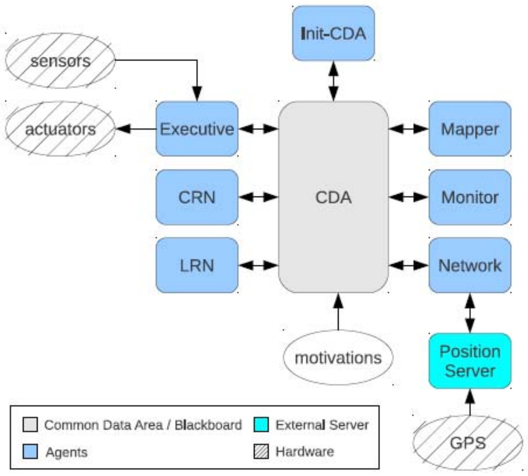


Figure 4. Blackboard-based system

The following is an abstract of the agents seen in Fig. 4:

- **Init-CDA:** initializes and then manages the CDA.
- **Network:** connects to a server process running on the experiment control system in order to receive position information from the GPS which it stores in the CDA. In our current experimentation, the GPS employs a video system to determine actual robot position using image analysis.
- **Executive:** responsible for executing motion commands, the interface between robot hardware and software, and the calculation of the actual position out of odometry and GPS data.
- **Mapper:** mapper based on the fuzzy ART algorithm [39].
- **CloseRangeNavigator (CRN):** responsible for navigating nearby obstacles. It is based on a fuzzy inference system (FIS), which converts sensor signals into actions and is completely reactive [40].
- **LongRangeNavigator (LRN):** is the planning module that plans the best route from one point to another. As described previously in Section 4.8, the calculated routes are determined by a GA depending on the motivation set M and the fitness scores F determined by the path taken by the robot agent through the robot's internal map (e.g. Fig. 7).
- **Monitor:** a two state finite state machine responsible for choosing the current navigation mode between LRN and CRN, this is currently wholly dependent upon potential collision hazard to the robot as measured by on-board infra-red (IR) sensors and a configurable proximity threshold value. The Monitor will terminate navigation if all missions have been achieved or when the maximum number of steps are completed.

4.7 Executive Implementation

The principal role of the Executive agent is to execute the motion command (i.e. speed and steering) chosen by the

Monitor agent. There could be multiple motion commands recommended by different agents (e.g. CRN, LRN) in the CDM, but the Monitor will choose only one which the Executive will transmit to the robot hardware.

Cumulative positioning errors are a well known issue in mobile robotics. These positioning errors may be due to motor noise, errors with the movement model used and positioning errors due to the external reference (e.g. GPS) itself. We tried to reduce model movement errors by using three bi-cubic spline movement models (i.e. $\gamma_1, \gamma_2, \gamma_3$) for the obtained variations in robot position (i.e. $\Delta X = \gamma_1(\mathrm{sp}, \mathrm{st}), \Delta Y = \gamma_2(\mathrm{sp}, \mathrm{st})$) and orientation (i.e. $\Delta\Theta = \gamma_3(\mathrm{sp}, \mathrm{st})$) after a motion command composed of normalized speed (sp) and steering (st) was selected by the Monitor agent. This approach was found to have reduced errors when compared with linear, quadratic and cubic least squares models [41]. As previously mentioned, in order to limit cumulative positioning errors the Executive agent will also periodically query current robot position and will make position corrections as needed.

4.8 LRN Implementation

The objective of the LRN agent is to perform path planning and to generate a sequence of highly fit actions (i.e. steps) for the robot. The LRN uses a GA to find the best possible route according to motivation settings and the chosen test environment. Each individual in the GA encodes all planned actions into its genome as a sequence and the best individual is chosen at the end of the optimization. Fig. 5 shows how the LRN performs path planning using fuzzy motivations in the context of the multiple agent architecture.

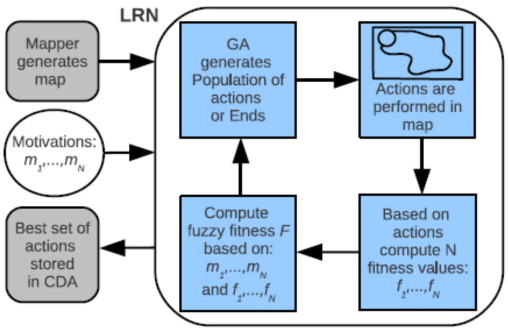

LRN
Mapper generates map
Motivations: $m_1,\ldots,m_N$
Best set of actions stored in CDA
GA generates Population of actions or Ends
Actions are performed in map
Based on actions compute N fitness values: $f_1,\ldots,f_N$
Compute fuzzy fitness F based on: $m_1,\ldots,m_N$ and $f_1,\ldots,f_N$

Figure 5. LRN overview

4.8.1 GA Parameters

Previous researchers have looked into the issue of determining GA parameters that will work relatively well in typical optimization problems. From [43] and [44], a typical parameter set that is recommended is: a population of 20 individuals, two point crossover, probability of crossover of 0.95, probability of mutation of 0.01 and 100 iterations. Additionally, a method of tournament selection with one elite individual is recommended.

These parameters work well in the context of function optimization and we have taken them into consideration as the starting point for determining reasonable parameters for the task at hand. This, along with the consideration of having a general set of robust GA parameters that will work well in a variety of problems, is an open research topic that will likely not be resolved in the near future.

In our route planning task, we wanted to have a system that performed the long range planning task in the order of several seconds at most. This required several compromises when considering the small CPU that was available in our embedded robotic context. With these facts in mind, and after extensive experimentation with the system, we settled on the following values for GA parameters: population size=30, crossover points=20, selection method=Tournament with Elite strategy, mutation rate=20%, generations=20.

We empirically found that having a greater mutation rate and a larger number of crossover points than was typically recommended allowed the population to continually explore the fitness landscape and find good solutions with a smaller number of generations (i.e. 20) than the previously expected value (i.e. 100). Clearly these solutions are not optimal in the navigation sense but allow for the robot to explore its environment (i.e. perform its long range and short navigation tasks adequately) within time constraint requirements. As described in the next section, the number of actions planned per LRN phase was constrained to 40 also with this goal in mind.

As seen in the examples in Fig. 6, with these settings the GA is capable of steadily increasing the average fitness of the best individual in the population. These parameter values represent a compromise between quality of solutions and processing speed. They result in a planning time of around 10 seconds and considering that it is running in a real-time embedded solution, this compromise was found to be reasonable. In the figure, we show normalized fuzzy fitness scores (to the maximum possible) averaged over ten runs. Please note that the average fuzzy score is low because in the case that an agent crashes its fitness is set to zero. This low average score is probably also partly due to the high mutation rate, but is a good trade-off considering that the best individual average fitness is always increased (i.e. given the elite strategy used). Finally, fitness results could be improved if the number of generations was increased, but this would result in worse real-time performance.

4.8.2 GA Encoding and Solution Search Space

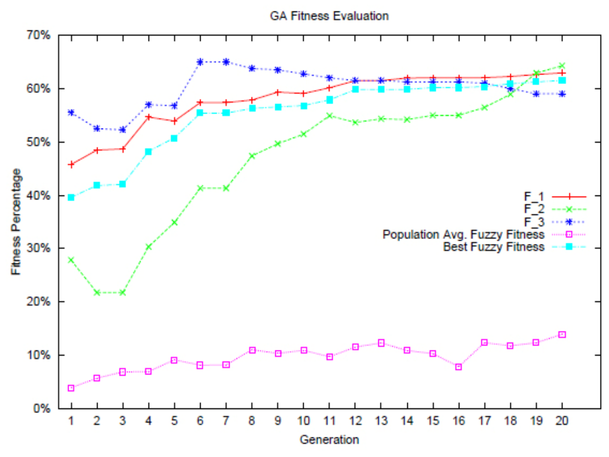
Considering real-time execution, we selected each gene in the individual path to consist of only one of three discrete actions: forward, turn right 30°, turn left 30°. Hence, the solution search space is 3^n. In order to reduce the search difficulty during route planning, the GA generates a maximum number of steps in each planning phase where the value of n is smaller than the number of steps in the total experiment. The GA process runs repeatedly until the total number of steps required by the experiment is completed. We empirically found 40 to be a reasonable compromise value for the number of actions planned in each LRN phase in terms of obtained path fitness and computational time required.

Using a number of actions per phase has an impact on the overall behaviours due to Curiosity, Homing, Energy or Mission motivations, as the behaviours per phase are dependent on previous ones. This is akin to performing multiple local searches over the overall search space which is a common mechanism in many well known algorithms (e.g. stochastic hill-climbing, simulated annealing, etc.). In the case of Homing, using this motivation and associated fitness in a per phase manner causes the robot to want to stay near home in each phase which is not exactly the same behaviour that might be expected over the entire experiment. This could have been resolved using a heuristic considering a threshold on the minimum % of the overall steps required before the Homing motivation was considered in planning and perhaps a graded response to the associated urgency (e.g. like the 1/3 spent air rule prior to forcing a return in the case of scuba cave divers). Note that this was not implemented in this instance in order to allow for a more straightforward comparison of agent behaviours.

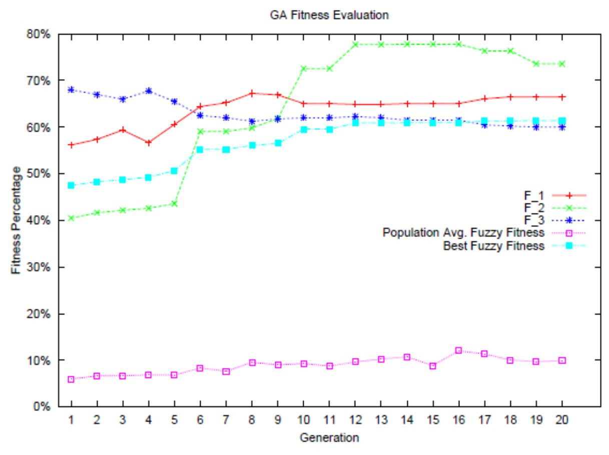
After each set of movements is executed in the internal map, a new planning phase starts from the current position (see Fig. 7 where each new planning phase is shown as a separate sub-figure. As seen in Fig. 7(h), in most actual cases the planned steps in each previous phase are not completely executed. This is due to two situations: because the Monitor process will switch execution to the CRN in case of a potential collision hazard to the robot or the current internal position error is greater than a threshold, in which case it is corrected. In either case, the LRN will enter a new planning phase and will generate a new route to be executed.

As previously mentioned, a route calculation takes ≈ 10 seconds during which time the robot does not move unless it is interrupted (e.g. by an object moving into its collision threshold range) in which case the Monitor has the CRN take over robot motion to allow it to escape danger.

Each agent is implemented as an independent process of the operating system so that the time given to it is managed by the operating system (OS). We have also utilized the micro sleep function of the OS in the task loop of each agent. This helps to insure access fairness so that all agents can participate of CPU time. Given this implementation, the LRN to CRN commutation takes ≈ 0.1 seconds which allows the robot to escape imminent danger. In this regard, the architecture allows an implementation that does not suffer from traditional cognitive overload because even though the LRN agent may be performing complex CPU consuming tasks, it is encapsulated to only one process and other agents can still operate in parallel.



(a) Curiosity=0.7, Homing=0.3



(b) Curiosity=0.3, Homing=0.7

Figure 6. GA average fitness evolution

5. Experimental Evaluation

We performed several robotic experiments including route planning and mission completion for various motivations sets, rooms and missions. All these experiments were also a test of whether the motivation-based interface could help drive and understand agent behaviour. Ten experiments were performed for each motivation setting and average results are provided.

As seen in Fig. 8, we have designed four different rooms with limits of 200 cm x 250 cm for the robot to navigate in. We denote these rooms as: R-ROOM, L-ROOM, T-ROOM and H-ROOM. The square in each room represents the robot starting zone (i.e. home neighbourhood).

5.1 Robot Description

We used a low cost robot to facilitate these experiments. It consists of: a Boe-Bot [45] chassis (≈ 200 mm in diameter), a Gumstix Verdex pro XM4-BT (400 MHz, 64 MB RAM, 16 MR Flash, Bluetooth interface) with a Robostix (ATMega128 based), five Sharp 2D120X IR sensors, and LM7805 voltage regulators.

We chose to have each forward action move the robot 9 mm, the total number of actions in the experiment was set to 1000 to allow for sufficient exploration of the test environments.

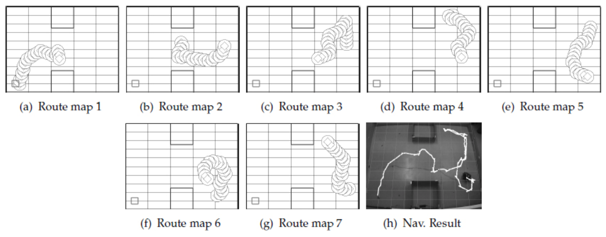


Figure 7. GA-based route planning and actual navigation (Curiosity=1.0), each route map corresponds to an LRN phase of 40 steps

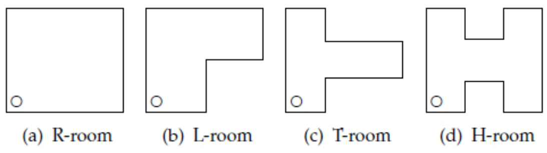


Figure 8. Experimental room layout

As part of robot movement calibration, we configured each action as the following number of consecutive motor commands:

- Forward: 7
- Turn right 30%: 6
- Turn left 30%: 6

Total battery budget for each experiment was set to 25 virtual energy units. Each forward motion used 0.025 energy units, and right and left movement energy costs were 0.0107 energy units due to having only one servo active during these actions and the duration being 6/7 of the forward motion.

5.2 Motivation Set

The fuzzy motivation profile (M) used included: Curiosity (m_1), Homing (m_2), Energy (m_3) and Mission (m_4). The Mission in these tests was to reach the target position in the upper left quadrant of the map (indicated as a crossbar), this is only applicable in the cases when $m_4 > 0$.

As presented in Fig. 9, the leftmost images have the highest motivations for Homing, Energy and Mission respectively, and the rightmost images have the highest for Curiosity.

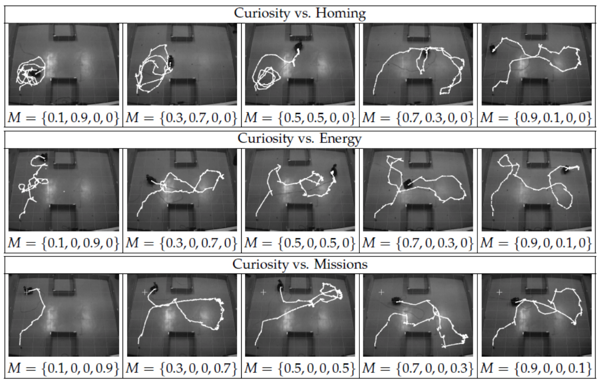


Figure 9. Robot navigation for different motivation settings (navigation ends at maximum number of steps or mission completion)

5.3 Results Evaluation

As seen in Figs. 10 - 12, we contrasted the resultant behaviours of the following motivation pairs: Curiosity vs. Homing, Curiosity vs. Energy and Curiosity vs. Missions. In order to evaluate navigation results, we utilized logging in the physical robot to measure movement selection and position related metrics (e.g. Battery %, Forward %, Last position %, Farthest position %, Accomplished %) and an external video system to evaluate other metrics (e.g. Exploration %):

- Forward %: this is the percentage of forward vs. lateral movements performed in the entire experiment,
- Battery %: this is the percentage remainder battery budget at the end of the experiment,
- Exploration %: this is the percent exploration over the entire area,
- Last position %: this is the last robot position over the maximum possible distance,
- Farthest position %: this is the farthest robot position over the maximum possible distance,
- Accomplished %: this is the percentage of missions that were accomplished.

5.3.1 Curiosity vs. Energy Evaluation

As seen by the results in Table 1 and Fig. 10, when Curiosity is increased vs. Energy, the robot tends to increase its forward motion and also explores a greater area of the map. Remainder battery also tends to decrease as a result of the additional forward motion.

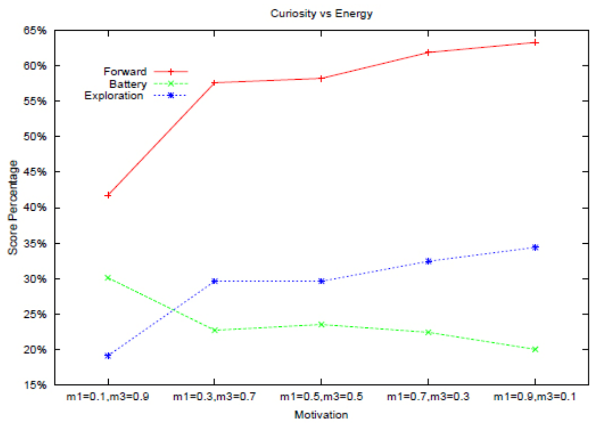


Figure 10. Robot behaviour evaluation of Curiosity vs. Energy

Motivation	Forward %	Battery %	Exploration %
m_1=0.1, m_3=0.9	41.7 ± 2.8	30.1 ± 2.1	19.1 ± 3.0
m_1=0.3, m_3=0.7	57.9 ± 5.3	22.7 ± 2.7	29.9 ± 4.6
m_1=0.5, m_3=0.5	58.5 ± 5.1	23.5 ± 2.8	29.9 ± 7.7
m_1=0.7, m_3=0.3	62.1 ± 5.9	22.4 ± 2.7	33.7 ± 8.5
m_1=0.9, m_3=0.1	63.3 ± 5.8	20.0 ± 2.6	34.4 ± 8.6

Table 1. Curiosity vs. Energy

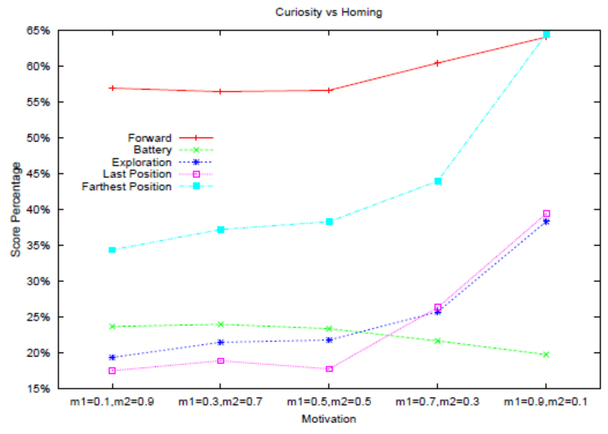


Figure 11. Robot behaviour evaluation of Curiosity vs. Homing

5.3.2 Curiosity vs. Homing Evaluation

In the results seen in Table 2 and Fig. 11, as Curiosity is increased vs. Homing, the metrics related to exploration increase. The final distance of the robot from its home location and the farthest position visited also follow this trend.

5.3.3 Curiosity vs. Missions Evaluation

In the results seen in Table 3 and Fig. 12, as Curiosity (vs. Missions motivation) is increased, the percent of accomplished missions decreases dramatically. As expected, exploration tends to increase, forward percentage tends to decrease and battery usage tends to decrease. Please note that the number of steps executed in all runs is not the same in all cases given that as previously described when the mission is accomplished the run ends. This would tend to explain the increase in exploration percentage for the case $m_1 = 0.7, m_2 = 0.3$.

Motivation	Forward %	Battery %	Exploration %
m_1=0.1, m_2=0.9	56.9 ± 3.2	23.7 ± 2.2	19.4 ± 8.9
m_1=0.3, m_2=0.7	56.4 ± 3.3	24.0 ± 2.3	21.5 ± 8.7
m_1=0.5, m_2=0.5	56.6 ± 3.3	23.4 ± 2.5	22.0 ± 8.0
m_1=0.7, m_2=0.3	60.4 ± 2.8	21.7 ± 2.2	25.3 ± 7.6
m_1=0.8, m_2=0.1	64.0 ± 2.5	19.8 ± 1.8	38.3 ± 5.8
	Last Pos.%	**Farthest Pos.%**	
m_1=0.1, m_2=0.9	17.6 ± 4.4	34.4 ± 6.1	
m_1=0.3, m_2=0.7	18.9 ± 4.7	37.2 ± 6.0	
m_1=0.5, m_2=0.5	17.8 ± 4.9	38.3 ± 6.2	
m_1=0.7, m_2=0.3	26.4 ± 5.5	43.9 ± 4.7	
m_1=0.9, m_2=0.1	39.4 ± 5.4	64.4 ± 4.4	

Table 2. Curiosity vs. Homing

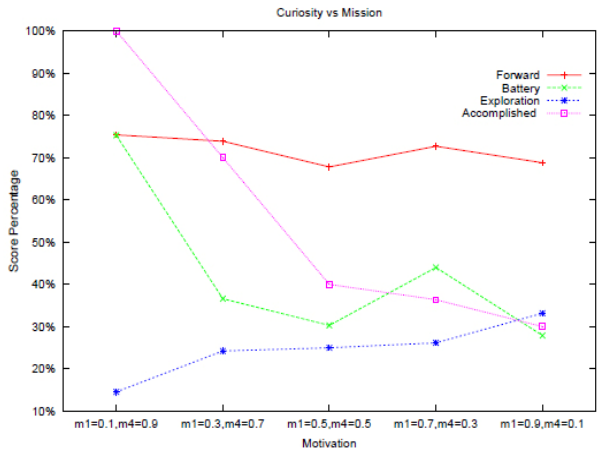


Figure 12. Robot behaviour evaluation of Curiosity vs. Missions

Motivation	Forward %	Battery %	Expl. %	Accomp.%
m_1=0.1, m_4=0.9	75.4 ± 23.3	75.3 ± 1.8	14.5 ± 4.3	100
m_1=0.3, m_4=0.7	73.9 ± 20.9	36.6 ± 1.9	24.9 ± 4.4	73
m_1=0.5, m_4=0.5	67.8 ± 19.3	30.3 ± 2.0	25.3 ± 3.8	36
m_1=0.7, m_4=0.3	72.7 ± 28.9	44.0 ± 2.5	25.3 ± 4.7	42
m_1=0.9, m_4=0.1	68.8 ± 25.1	27.9 ± 2.5	32.9 ± 4.8	36

Table 3. Curiosity vs. Missions

6. Conclusions

The scheme of behaviour learning using fuzzy motivations was successfully applied in a low cost robot with a blackboard-based multiple agent architecture. This biologically-based approach was validated as the implemented mobile robot developed a variety of behaviours in accordance with user specified fuzzy motivation values. We conclude that fuzzy motivations present a simple interface for end user robot management.

The multi-agent blackboard architecture supported the incremental building of complexity into the robot. We introduced different navigation strategies: reactive, planned and feature-based. The first navigation process implemented was the close range navigator (CRN), followed by the long range navigator (LRN) and then a feature navigator (not part of the current investigation). Our implementation of this architecture did not evidence cognitive overload as the Monitor agent (and the robot) remained responsive to threats (with the CRN) in the midst of intensive calculations by the LRN.

There are several trade-offs in time vs. performance that had to be considered during tuning of robot parameters. This parameter tuning process resulted in an acceptable robot performance in this CPU constrained context, but should improve in systems with greater processing power. This suggests that this implementation would work well with a variety of platforms and applications.

Future work includes adding further agents (e.g. cognitive agents) to complement those currently in the system, further experimentation regarding feature recognition, testing other mapping capabilities and adding basic object manipulation. This project is open-source and the complete source code is available in [46].

7. Acknowledgements

This research was partially funded by the research authority (DGIP) of UTFSM (231356). We would like to thank Linus Casassa and the Electronic Dept. Workshop for their help with robot construction. We also thank Gabriel Cisternas, Benjamin Almarza and Felipe Conejera for their help on associated empirical studies. Finally, we would like to thank Ethel Vidal H. and Prof. Fernando Auat Cheein for reviewing this paper.

8. References

[1] Reiss, S.: Multifaceted Nature of Intrinsic Motivation: The Theory of 16 Basic Desires, Review of General Psychology, Vol. 8, No. 3, 2004, pp. 179-193.

[2] Huitt, W.: Motivation to learn: An overview. Educational Psychology Interactive. Valdosta State University.: http://chiron.valdosta.edu/whuitt/col/motivation/motivate.html, (2001).

[3] Woolridge, M.: An Introduction to Multi Agent Systems, Wiley, West Sussex, England, 2002.

[4] Park, H., Kim, E., Kim, H.: Robot Competition Using Gesture Based Interface. In: Moonis, A. Esposito, F. (eds): Innovations in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 3353. Springer-Verlag, Berlin, 2005, pp. 131-133.

[5] Tasaki, T., Matsumoto, S., Ohba, H., Toda, M., Komatani, K., Ogata, T., Okuno, H.: Distance-Based Dynamic Interaction of Humanoid Robot with Multiple People. In: Moonis, A. Esposito, F. (eds): Innovations in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 3353, Springer-Verlag, Berlin, 2005, pp. 111-120.

[6] Jensen, B., Tomatis, N., Mayor, L., Drygajlo, A., Siegwart, R.: Robots Meet Humans Interacion in Public Spaces, IEEE Transactions on Industrial Electronics, Vol. 52, No. 6, 2006, pp. 1530-1546.

[7] Arkin, R.: Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.

[8] Arredondo, T., Freund, W., Muñoz, C., Navarro, N., and Quirós, F.: Fuzzy Motivations for Evolutionary Behavior Learning by a Mobile Robot. In: Moonis, A., Esposito, F. (eds): Innovations in Applied Artificial Intelligence, LNAI, Vol. 4031. Springer-Verlag, Berlin, 2006, pp. 462-471.

[9] Brooks, R.: A Robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, 1986, pp. 14-23.

[10] Davis, D.N., Gwatkin, J.: robo-CAMAL: A BDI Motivational Robot, Journal of Behavioral Robotics, 1(2), 2010, pp. 116-129.

[11] Arredondo, T.: Fuzzy Motivations in Behavior Based Agents. Smart Information and Knowledge Management, Vol. 260. Springer-Verlag, Berlin, 2010, pp. 247-272.

[12] Agile Development at the Wikipedia website: http://en.wikipedia.org/wiki/Agile_software_development

[13] Iterative Development at the Wikipedia website: http://en.wikipedia.org/wiki/Iterative_and_incremental_development

[14] Bratman, M.: Plans and Resource-Bounded Practical Reasoning. Computational Intelligence, 4(4) (1988) 349-355

[15] Rao, A.S., Georgeff, M.P.: BDI Agents From Theory to Practice. Proceedings of the First International Conference on Multi Agent Systems (1995)

[16] Georgeff, M.P., Ingrand, F.F.: Decision-Making in an Embedded Reasoning System. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (1989)

[17] Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence, 69 (1994) 165-204

[18] Norman, T.J.: Motivation-based direction of planning attention in agents with goal autonomy, PhD Thesis, Department of Computer Science, University College of London, UK. (1997)

[19] Coddington, A.M, Luck, M.: A Motivation Based Planning and Execution Framework. International Journal on Artificial Intelligence Tools, 13(1) (2004) 5-25

[20] Nourbakhsh, I., Bobenage, S., Grange, S., Lutz, R., Meyer, R., Soto, A.: An Affective Mobile Educator

with a Full-time Job, Artificial Intelligence, 1-2(114), 1999, pp. 95-124.

[21] Michaud, F., Vu., M.: Managing robot autonomy and interactivity using motives and visual communication, Proceedings of the Third Annual Conference on Autonomous Agents, Seattle, WA, 1999, pp. 160-167.

[22] Breazeal, C., Velasquez, J., Robot in Society: Friend or Appliance? In Proceedings of Agents 99 Workshop on emotion-based agent architectures, Seattle, WA, 1999, 18-26.

[23] Hirth, J., Berns, K.: Emotion-based Architecture for Social Interactive Robots, Humanoid Robots, Ben Choi (Ed.), InTech, Slavka, Croatia, 2009.

[24] Stoytchev, A., Arkin, R.: Incorporating Motivation in a Hybrid Robot Architecture, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.8, No. 3, 2004, pp. 100-105.

[25] Cloos, C., The Utilibot Project: An Autonomous Mobile Robot Based on Utilitarianism, 2005 AAAI Fall Symposium on Machine Ethics, AAAI Technical Report FS-05-06, 2005, pp. 38-45.

[26] Jang, J., Chuen-Tsai, S., Mitzutani, E.: Neuro-Fuzzy and Soft Computing, Prentice Hall, NJ, (1997).

[27] Karray, F.O., De Silva, C.: Soft Computing and Intelligent Systems Design, Addison Wesley, Essex, England, (2004).

[28] Passino, K.: Biomimicry for Optimization, Control and Automation, Springer-Verlag, London, (2005).

[29] Zadeh, L. A.: "Fuzzy Logic, Neural Networks, and Soft Computing" Communications of the ACM, March 1994, Vol. 37 No. 3, pages 77-84.

[30] Konolige, K., Meyers, K., Saffiotti, A.: FLAKEY, an Autonomous Mobile Robot. SRI technical document, July 20 (1992).

[31] 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

[32] 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

[33] Hoffman, F.: Soft computing techniques for the design of mobile robot behaviors. Information Sciences, 122 (2000) 241-258

[34] Zhou, C.: Robot learning with GA-based fuzzy reinforcement learning agents, Information Sciences, 145 (2002) 45-68

[35] 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

[36] Martinez Barbera, H., Gomez Skarmeta, A.: A Framework for Defining and Learning Fuzzy Behaviours for Autonomous Mobile Robots, International Journal of Intelligent Systems, 17(1) , (2002) 1-20

[37] 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

[38] Eskridge, B., Hougen, D.: Extending Adaptive Fuzzy Behavior Hierarchies to Multiple Levels of Composite Behaviors, Robotics and Autonomous Systems, V.58 (2010) 1076-1084

[39] Araujo, R., de Almeida, A.: Fuzzy ART Based Approach for Real-Time Map Building, Proceedings of the IEEE 5th International Workshop on Advanced Motion Control, Coimbra, Portugal, 1998, pp. 623-628.

[40] Makoto, K., Peng-Yung, W.: Implementation of a Hexapod Mobile Robot with a Fuzzy Controller, Robotica, 23(6), 2005, pp. 681-688.

[41] Cisternas, G.E.: Investigación e implementación de métodos basados en softcomputing para generación de mapas y navegación robotica, Undergraduate Thesis, Electronics Department, UTFSM, Chile. (2009)

[42] Maslow, A.H.: A Theory of Human Motivation, Psychological Review, 50(4), 1943, pp. 370-396.

[43] De Jong, K.A.:An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, Ann Arbor, MI, USA, 1975.

[44] Grefenstette, J.: Optimization of control parameters for genetic algorithms, IEEE Trans. Syst. Man Cybern., vol. 16, pp. 122-128, January 1986.

[45] http://www.parallax.com/go/boebot

[46] http://mari-robot.sourceforge.net/