Adaptive fuzzy knowledge-based systems for control metabots' mobility on virtual environments

A. Arroyo, F. Serradilla and O. Calvo

Dpto. Sistemas Inteligentes Aplicados, Universidad Politecnica de Madrid, Madrid, Spain
Email: aarroyo@eui.upm.es

Abstract: *The confluence of three-dimensional (3D) virtual worlds with social networks imposes on software agents, in addition to conversational functions, the same behaviours as those common to human-driven avatars. In this paper, we explore the possibilities of the use of metabots (metaverse robots) with motion capabilities in complex virtual 3D worlds and we put forward a learning model based on the techniques used in evolutionary computation for optimizing the fuzzy controllers which will subsequently be used by metabots for moving around a virtual environment.*

Keywords: knowledge-based systems, fuzzy control, evolutionary computation, intelligent agents, metabots, virtual worlds, second life, metaverses

1. Introduction

We are experiencing the development of so-called Web 2.0 and a large number of applications are being introduced into many users lives and are profoundly changing the roots of society by creating new methods of communication and cooperation. Similarly, in recent years, the first virtual worlds have emerged in which humans, through their avatars, 'cohabit' with other users. This is a new interaction model based on technology arising from multiplayer online games, a three-dimensional (3D) virtual world imagined by Neil Stephenson in his novel 'Snow Crash' (Stephenson, 1992) and previously by Vernor Vinge in the short story 'True Names' (Vinge, 1981), a world in which humans feel at home, a metaverse. This new model is more human because it simulates the real environment's characteristics in which the human being exists and has grown. If our senses create the reality in which we are developing ourselves, to experience virtual reality worlds we will have to live in accordance with the capabilities of our senses, to live in digital extensions of our physical world.

These virtual worlds, or metaverses, are in fact true social networks, and they are useful for interaction between people in different locations. Likewise, in the 3D context, we propose the development of virtual robots with the same appearance as human-driven avatars. To these new virtual robots we assign the term 'metabot' – a term coined from the contraction of the terms metaverse and robot. A metabot will therefore be a fully capable software element designed to interact in one or more metaverses through one or more avatars. The representation of a metabot should be much more intuitive to humans than the traditional one, as it simulates reality and therefore does not require any specific learning.

The development of these metabots involves many fields of research (conversational interaction,

emotional nature, social behaviour, mobility, etc.). In previous works, we have explored the conversational possibilities of metabots (Griol *et al.*, 2009, 2010a, 2010b), the space-time representation of their activities in the virtual environment (Arroyo *et al.*, 2010) and the many different means of intelligent interaction with users (Arroyo *et al.*, 2009). This article focuses on the mobility of the various types of metabots that we can implement in the virtual environment used in our experiments [Second Life (SL)] and we will show the results obtained by applying a learning model based on Evolutionary Computation which allows us to adjust the parameters of the Fuzzy Knowledge-Based Systems (FKBS) used in modelling the movement of the metabots.

Firstly, we present our research and related works. Secondly, we present the software substrate developed to support these new experiences. We then describe the fuzzy control system designed to control the movement of the different metabots and the evolutionary model used for its adaptation to our virtual training space (VTS) in SL. Finally, we discuss our experiments with metabots in detail basing our discussion on the two projects which have partially supported this research: (a) BABYBOT: Creating a 'virtual child' as a conversational agent. Artificial intelligence techniques for the detection of criminal behaviour in virtual environments and social networking; and (b) the 2008 and 2010 UPM competition calls. In the first case, we used the point of maximum attention of the most receptive avatar to place the virtual child in the best possible position for achieving its objectives, and in the second case, we defined complex mobility behaviour for predators and prey by merging the effects obtained by the different fuzzy controllers designed in the virtual competition.

2. Related works

The theoretical subject of study of Artificial Intelligence (AI) is cognition, the processes that allows a being to perceive what is happening on its environment, to reason and to act in this same environment. In the other hand, the practical purpose of AI is the construction of artificial beings with cognitive abilities, beings who can communicate with humans in natural language and can perform functions that require cognition in a given environment. Clearly, the most interesting experimental field is the natural environment in which we humans live. Within this environment we find Robotics as the main scope of theoretical cognition focused studies. The current metaverse, in its many forms (games and virtual worlds), provides us alternative time-space environments, 'realities' in which beings can be and identify being acts. Spaces in which human-driven avatars can coexist with autonomous avatars. Marvelous spaces for experimentation in models and Embodied Intelligent Agents architectures and multiagent systems with these characteristics and whose results are applicable to real world.

In 3D virtual spaces in which users establish contact through avatars that represent them, intelligent agents must show active behaviour. This behaviour (gestures, movements, etc.) has to emulate a human-driven avatar and reflect the personality of the profile or type of user it represents. The movements of the robots that inhabit 3D environments are based on the same principles as those used in mobile robotics for decades, so the models used in this field (Breazeal, 2003) are easy to implement in the simulations conducted in virtual worlds.

The fuzzy controllers designed for automating the movements of various different autonomous systems (cars, helicopters, mobile robots, etc.) have already been used successfully for quite some time. The results obtained by different researchers (Driankov & Saffioti, 2001; Vachkov & Fukuda, 2001; Cordón *et al.*, 2002; Pham *et al.*, 2007; Ross, 2007; Ho *et al.*, 2008; Iida *et al.*, 2009) have led to this model becoming one of the most effective in this field. For this reason, we decided to use it in preference to other possibilities on offer which we had used previously (Patricio *et al.*, 2005).

After years of research, fuzzy controllers have been proved as much more robust than the

traditional control on non-linear problems, and especially easier to build, making it suitable for use in conditions in which the traditional approach is too complex or hard to implement. There are numerous tasks solved using these controllers, as steering wheel control to track a beacon obtained by mapping and GPS (Rodríguez-Castaño *et al.*, 2000), Brake Assist (Kim *et al.*, 1996), maintaining a safe distance from the leading vehicle (Chen *et al.*, 2007), performing maneuvers (Naranjo *et al.*, 2008) or follow the preceding vehicle (Holve *et al.*, 1995), among others. Also investigated in the construction of controllers for complex control vehicles, as helicopters (Sugeno *et al.*, 1995) and quadricopters (Chriette *et al.*, 2009).

The Metaverse research has focused on the construction of Fuzzy Cognitive Maps (Parenthoën *et al.*, 2001) for modeling virtual worlds (Dickerson & Kosko, 1993). Mobility of avatars within the metaverse opens a brand of new possibilities, as determining the centre of attention in a group or evaluation or the popularity of a particular region.

One of the main problems with fuzzy controllers is the adjustment of fuzzy sets which must be made for each linguistic variable (LV). According to a large number of studies (Ching-Chang *et al.*, 2007; Kroeske *et al.*, 2008; Herrera & Lozano, 2009), evolutionary computation has proved itself to be an effective tool for resolving this important design flaw of the FKBS. In addition, this way of thinking has already been applied in previous developments as a learning model (Usero *et al.*, 2007).

Obviously, our case is especially suited to the use of evolutionary techniques as we are working in synthetic worlds in which it is possible to create and erase a large number of copies and to evaluate their behaviour in the same space in which they will go on to develop. Normally, in the real world, this process is much more expensive as it is necessary to design and build a simulator, and even in the best possible scenario it is only possible to achieve an approximation of the real environment in which the controllers will eventually be used.

3. Concepts on metabots

A metabot is a software robot that develops within a metaverse. Metabots can be just as external to the metaverse as humans are, and even the code which forms them may be run in another system or they can be built using the same material of the metaverse itself. They represent a new type of interface which is much more intuitive to humans, as they must be created following the principle of maximum verisimilitude with the real world and in particular with human anthropomorphism. In this respect the human–machine interaction is intuitive and does not require any special learning. From a human point of view a metabot is just another avatar or being with which we interact in conversation, and from the point of view of the machine it is a knowledge-based multi-agent system which indiscriminately uses any data channel such as information retrieval systems, ontologies, databases, indexing engines, social networks, recommender systems, public information, etc.

A metabot is also a mobile, proactive and perceptive agent. It must be capable of moving within a spatially immersive virtual environment in a similar way to human beings who experience the same environment. It must also be able to communicate with other avatars, whether these are controlled by metabots or humans, although with humans it should preferably use conversational systems. In the same way, in order to be able to interact in persistent complex 3D environments it must be supported by ontologies which create an anthropomorphic knowledge base in common with that of human beings.

Metabots can be of many different types; however they obey a very specific classification depending on the technological environment that we use for implementing them. On the one hand we have intra-metabots (abbreviated to iMetabots) which are constructed using the resources of the metaverse itself, and exo-metabots (or eMetabots, as they are referred to hereafter) which are external to the metaverse and connect to it in the same way as a human being does, through an avatar.

Although both types are comparable in functional terms, the difference in their conception means that each have their own different strong points.

For example, iMetabots, because they are immersed in the metaverse, have much faster physical reactions as they are part of the same process that is being simulated by that region and, as a result, they share memory, processor and data with the rest of the environment. However, this means that they must be implemented in LSL (the scripting language used in SL) and they have many limitations relating to their short assigned memory and processor.

On the other hand, eMetabots have much greater power in general. An eMetabot runs on a machine which is external to the metaverse, and consequently the limitations of memory and the processor capacity are only established by our own resources. However, an eMetabot reacts more slowly than an iMetabot as it must seek information from the environment and communicate its actions to the simulator process of the region.

3.1. iMetabot architecture

The main characteristic of an iMetabot is that it is designed using LSL. This requires us to know the process model and the platform where the metaverse runs in order to understand an iMetabot's architecture correctly.

As we are defining 'metaverse' in this article, and taking it into account that experiments are performed on SL, a region is an instance of an application known as a 'simulator'. This region is normally a 3D virtual space with sides measuring 256×256 m and having an indefinite height. Within this space there are objects with their own physical characteristics (shape, texture, mass, speed, etc.) and avatars, which make up the environment with which users and Metabots interact. The simulator is responsible for achieving a simplification of the Newtonian laws of physics using a physics engine. As well as all of these ordinary tasks, it also has another very important job: the execution of scripts.

Once this is understood, we can view an iMetabot as a set of physical objects which are imbued with a series of LSL scripts which enable it to interact with the environment of its simulator, whether they are objects or avatars performing a high-level function. All of these interactions are carried out through calls to the LSL library functions.

In our case, making use of the characteristics of LSL, the architecture of an iMetabot is built of various distinct blocks, each one implemented in an independent script and related to each other through the sending of internal messages using the llMessageLinked function. Each module contains an independent functional capability which can be one of the fuzzy controllers which we will discuss later in the article. Each block forms an interaction system as it perceives its environment through captured events and is capable of acting in it.

Each of the controllers implemented sends its result to the corresponding blocks depending on whether a change in the speed, height or angle of rotation of the object is required as it moves through the 3D virtual world, and these building blocks send the final value to be applied in each of these concepts to the Movement Applicator module.

3.2. eMetabot architecture

The architecture of an eMetabot as we have developed it has two fundamental parts: BotBridge and MetaBotLib. The BotBridge is responsible for achieving the logical connection of the eMetabot with the metaverse to which it wants to connect, regardless of who is offering this service. At the moment an eMetabot can connect to either SL or to Open Grid (Figure 1).

MetaBotLib is, as the name itself suggests, a Java library which greatly facilitates the task of creating an eMetabot from scratch. Simply defining a class which inherits from the MetaverseRobot abstract class will give us an operational eMetabot.

EMetabots have access to a great variety of functions in the MetaBotLib which enables them to interact with the metaverse and make

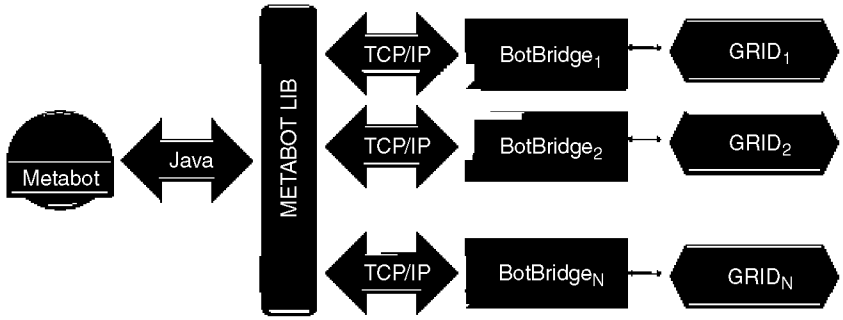

Figure 1: *The architecture of an eMetabot.*

use of all of its characteristics, but in addition, and as is logical, they can also use any existing Java library such as jFuzzyLogic, ecj, lucene, JDBC, etc., which turns them into potential multi-agent systems. It is even possible to implement an eMetabot which can control several avatars simultaneously or which can offer its services through several different channels: database, avatar, web server, web agent, etc. For example, an eMetabot which controls access to an area of the metaverse can accept HTTP requests with an ad-hoc web server in order to configure who has or does not have access. Another example is an eMetabot which can use Lucene to index all of the information it finds in the metaverse in order to provide a classic contents search.

MetaBotLib is a software platform designed to allow us to develop eMetabots in Java. Just creating a new object inheriting all the properties of a generic eMetabot we will have a full functional program that runs in SL in the shape of a normal Avatar. This API provides a set of methods and properties which makes our eMetabot able to perform any possible action within this metaverse: to walk, to fly, to listen, to speak, etc. This platform was developed by our team in the early phases of research and quickly became the common base of all developments with eMetabots. This piece of software allow us reuse several useful functionalities, saving hours of interface coding letting us centering on the algorithms and techniques which are the goal of our research.

4. Adaptative Fuzzy KBS for control metabots' mobility

For performing mobility actions with metabots we need a system which is capable of avoiding obstacles. To this end we have developed a movement control system for metabots which is based on FKBS. Within these FKBS there are two types, known as the Mamdani type FKBS and the TSK (Takagi, Sugeno and Kang) which differ mainly in the inference method used (Ross, 2007). The Mamdani model is characterized by its ease of interpretation, and the TSK model is characterized by its precision. The rules in the Mamdani model are in the format: IF (x is A) THEN (y is B), whereas in the TSK model they take the format: IF (x is A) THEN ($y = f(x)$). We use the Center of Gravity (CoG) to calculate the output value in the defuzzification process, a Mamdani type inference based on the composition of partial outputs, and our knowledge base is composed of all the possible combinations of the input variables for each of the controllers developed.

In our case we use the Mamdani model, mainly because of the easier legibility of its set of rules and because, in tests carried out, we have no need for greater computational efficiency, nor do we need to guarantee continuity of the output (SL has its limitations and it makes no sense to apply improvements if these are not useful in the execution environment).

When it comes to designing a fuzzy controller, we need to define the LV which must be taken

into account as input and output variables for our controller. Each LV is characterized by LV = {*X*, *T*, *U*, *M*} where *X* = name of the LV, *T* = set of linguistic tags, *U* = command of the LV and *M* = group of membership functions which associate each linguistic tag with a fuzzy set. In designing the fuzzy controller it is impossible to determine *a priori* the most suitable membership functions for its correct functioning. As we have already mentioned, evolutionary computation has proved to be a very useful tool in resolving this problem.

4.1. *Example for iMetabots*

This example demonstrates a speed controller, which depends on the number of obstacles detected in the direction of movement to be applied along with all of the other controllers responsible for guiding the iMetabot towards its objective, adapting its speed to the complexity of the environment in which it is moving. In this case, we decided to use Gaussian functions, allowing the evolutionary process to take care of adapting the points which define these functions (the average and standard deviation). Each LV (both in terms of input and output) is coded with an array of three genes, and each gene has a length of 4 bits which code the Gaussian membership function:

$$f(x) = ae^{-\frac{(x\ b)^2}{2c^2}}$$

where a is always 1 (the standard typical value in membership functions), $b = \mu$ and $c = \sigma$, and where the first two bits of code identify the offset from the mean ($\Delta\mu$) and the last two identify the offset from the standard deviation ($\Delta\sigma$).

As a result, a possible genome for the variable speed could be 0001 0100 1110, where each group of 4 bits corresponds to one of the membership functions represented in Figure 2.

The range of a variable is calculated as range = 1.1(max − min) to enable the membership functions to include extreme values. The speed variable has a minimum value of 0 m/s and a maximum value of 5 m/s, with the result that the range of this variable is 5.5. The values of μ and σ are calculated using the following expressions:

$$\mu = \left(posGene\frac{Range}{numGenes}\right) + \left(\Delta\mu\frac{Range}{sizeGenome}\right)$$

$$\sigma = (\Delta\sigma + 1)\frac{2\,Range}{sizeGenome}$$

In the case in question, for the first gene (0001), $\Delta\mu = 00 = 0$ and $\Delta\sigma = 01 = 1$, and as such, applying the expressions shown above, this will code a Gaussian membership function where $\mu = 0$ and $\sigma = 1.83$; for the second gene (0100), $\Delta\mu = 01 = 1$ and $\Delta\sigma = 00 = 0$, with the result that it will code a Gaussian membership function where $\mu = 2.29$ and $\sigma = 0.92$, and for the third gene (1110), $\Delta\mu = 11 = 3$ and $\Delta\sigma = 10 = 2$, which will code a Gaussian membership function where $\mu = 4.58$ and $\sigma = 2.75$ (as shown in Figure 3).

Figure 3 shows the whole fuzzy controller expressed in Fuzzy Control Language (FCL) for the individual with the best fitness obtained after 100 generations (with 500 individuals in the population), and the set of all of the input and output variables for the controller as well as an example of the output obtained (− 1.07) by this controller for an input speed of 2.7 m/s and a total of seven obstacles located by the sensor.

4.2. *Example for eMetabots*

This example shows a rotation controller based on the direction of the resultant force obtained from the equations in Serradilla & Gómez-Allende (1997) and on the actual direction of the eMetabot. In order to simplify its design and optimization, we used a single input variable calculated as the difference between these two directions expressed as an absolute value. It is possible to use this simplification as we are working on the assumption that rotations in SL will display the same behaviour whether the turn is made to the left or right. As a result, our input variable will have a range of values between 0 and Π. The output variable represents the application time of the rotational command to MetaBotLib. As such, it indicates the number of seconds that the command must remain active (simulating the length of time that a user keeps

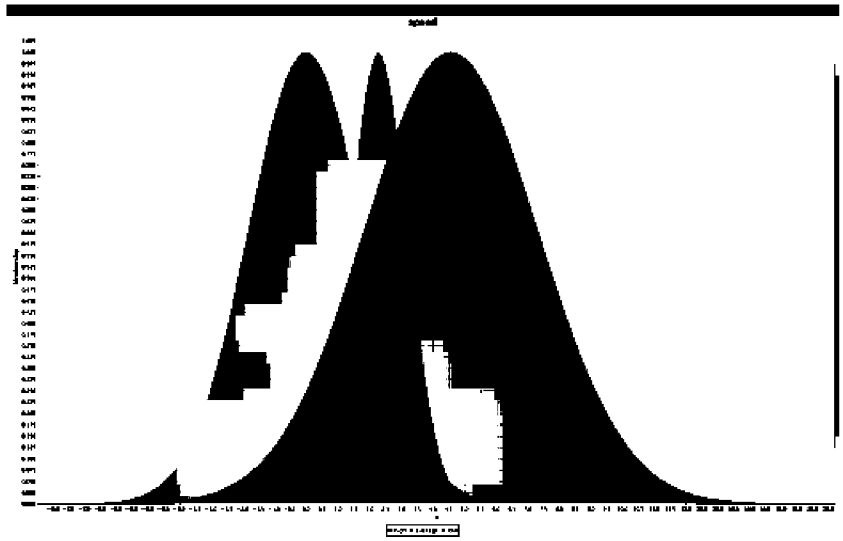

Figure 2: *The first group (first gene) of 4 bits (0001) corresponds to the linguistic tag 'low' in blue, the second (0100) to the tag 'average' in red, and the third (1110) to 'high', in grey.*

```
FUNCTION_BLOCK SpeedController2AvoidObstacles

VAR_INPUT
        numObst : REAL;
        speed : REAL;
END_VAR

VAR_OUTPUT
        action : REAL;
END_VAR

FUZZIFY numObst
        TERM average := GAUSS 6.5 2.5;
        TERM few := GAUSS 0.5 1.8;
        TERM many := GAUSS 9.2 1.9;
END_FUZZIFY

FUZZIFY speed
        TERM average := GAUSS 2.29 0.92;
        TERM high := GAUSS 4.58 2.75;
        TERM low := GAUSS 0.0 1.83;
END_FUZZIFY

DEFUZZIFY action
        TERM accelerate := GAUSS 2.1 1.5;
        TERM brake := GAUSS -2.4 2.2;
        TERM equal := GAUSS 0.0 1.6;
        METHOD : COG;
        DEFAULT := 0.0;
        RANGE := [-11.2000000000000001 .. 8.3];
END_DEFUZZIFY

RULEBLOCK Rules
        ACT : MIN;
        ACCU : MAX;
        AND : MIN;
        RULE 1 : IF (speed IS low) AND (numObst IS few) THEN action IS accelerate;
        RULE 2 : IF (speed IS low) AND (numObst IS average) THEN action IS accelerate;
        RULE 3 : IF (speed IS low) AND (numObst IS many) THEN action IS equal;
        RULE 4 : IF (speed IS average) AND (numObst IS few) THEN action IS accelerate;
        RULE 5 : IF (speed IS average) AND (numObst IS average) THEN action IS equal;
        RULE 6 : IF (speed IS average) AND (numObst IS many) THEN action IS brake;
        RULE 7 : IF (speed IS high) AND (numObst IS few) THEN action IS equal;
        RULE 8 : IF (speed IS high) AND (numObst IS average) THEN action IS brake;
        RULE 9 : IF (speed IS high) AND (numObst IS many) THEN action IS brake;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

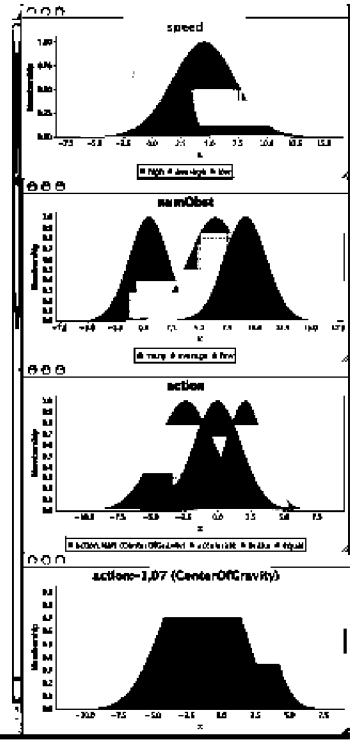

Figure 3: *Speed controller based on the number of obstacles detected during object movement.*

the corresponding key pressed down). We have observed empirically that a length of time of one second is more than sufficient for an avatar to make a half turn, which has enabled us to establish the range of the output variable as being between 0 and 1. Selection of the rotation command to be applied is external to the controller and only depends on the symbol of the difference between both directions.

As in the example given in the previous section, with this controller we used Gaussian functions to define the membership functions for the linguistic sets used. However, in this case, because we were using symmetrical values, we decided to extend the range of sets used from 3 to 5 (very little, little, average, much and very much) with the result that the chromosomes representing these increased from 12 to 20 bits.

Figure 4 shows the whole fuzzy controller expressed in FCL for the individual with the best fitness obtained after 50 generations (with 60 individuals in the population) and which corresponds to the chromosome [0100 0010 1001 0101 1110] for the input variable and [0000 0001 0101 0100 1000] for the output variable, and the input and output variables for the controller as well as an example of the output obtained by this controller (0.36) given an angle of 1.23 radians in the difference between the resultant obtained and the actual direction of the eMetabot.

5. Experimentation on metabots

The following sections describe the experiments carried out with both methods within the projects which have partially supported these works. We used Java open coding tools both for implementing the fuzzy controllers

```
End FruiSering
"Cimac-de-angel-arroyo-castillo-4:jFuzzyLogic angel$ java ExpertSystems/RotObstEvEMetabot
#
Begin RotObstEvEMetabots
FUNCTION_BLOCK RotationController2AvoidObstaclesAvXMetabots

VAR_INPUT
        diffResAndCurrentDirection : REAL;
END_VAR

VAR_OUTPUT
        turn : REAL;
END_VAR

FUZZIFY diffResAndCurrentDirection
        TERM average := GAUSS 1.728 0.691;
        TERM little := GAUSS 0.691 1.037;
        TERM much := GAUSS 2.246 0.691;
        TERM veryLittle := GAUSS 0.173 0.346;
        TERM veryMuch := GAUSS 3.283 0.346;
END_FUZZIFY

DEFUZZIFY turn
        TERM average := GAUSS 0.495 0.22;
        TERM little := GAUSS 0.22 0.22;
        TERM much := GAUSS 0.715 0.11;
        TERM veryLittle := GAUSS 0.0 0.11;
        TERM veryMuch := GAUSS 0.99 0.11;
        METHOD : COG;
        DEFAULT := 0.0;
        RANGE := (-0.66 .. 1.43);
END_DEFUZZIFY

RULEBLOCK Rules
        ACT : MIN;
        ACCU : MAX;
        AND : MIN;
        RULE 1 : IF diffResAndCurrentDirection IS veryLittle THEN turn IS veryLittle;
        RULE 2 : IF diffResAndCurrentDirection IS little THEN turn IS little;
        RULE 3 : IF diffResAndCurrentDirection IS average THEN turn IS average;
        RULE 4 : IF diffResAndCurrentDirection IS much THEN turn IS much;
        RULE 5 : IF diffResAndCurrentDirection IS veryMuch THEN turn IS veryMuch;
END_RULEBLOCK

END_FUNCTION_BLOCK


End RotObstEvEMetabots
```

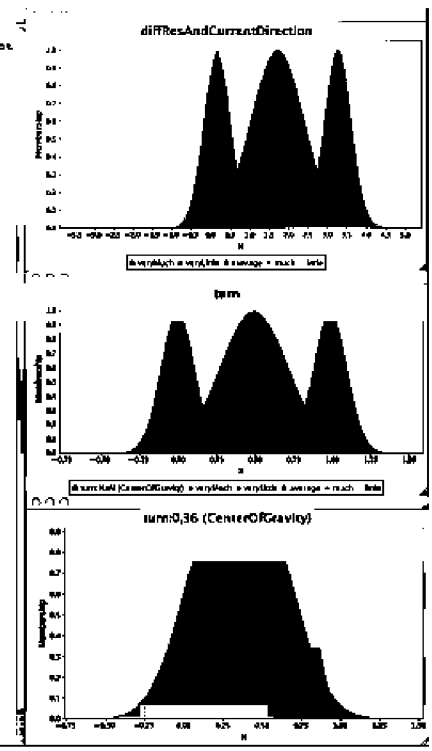

Figure 4: *Rotation controller based on the difference between the resultant value and the actual direction of the eMetabot.*

(Cingolani, 2005) and for learning through evolutionary computation (Luke *et al.*, 2009).

5.1. UPM competition calls

The 2008 competition (http://aicu.eui.upm.es/sl/doku.php) was designed as a classic predator–prey type problem, in which the bots (both predators and prey) move around the aquatic environment on our virtual island. We designed our prey according to the method presented in this article, and the teams of students entered in the competition had to design the predators, which had the objective of locating and making contact with the prey (Figure 5).

The participants had to design or programme an object or group of objects with the ability to move around the aquatic environment on the virtual island of TESIS while avoiding static or moving obstacles which they might meet along the way. As they moved around they had to catch the greatest possible amount of prey by making physical contact with it. The greater the 'intelligence' of the prey caught, the greater the number of points this contact was worth. The teams could design their strategy freely using a single object or a group of objects, but with the restrictions that, whatever their choice, the maximum number of objects could never exceed ten, the total number of *prims* used had to be < 300, they would have a maximum speed of 5 m/s and the total volume occupied by all of the objects for the team could not be greater than 3 cubic metres.

In our case, we had to control height, speed and direction of movement, all the while taking into account the objective of the specific prey bot that we were designing. As they were multiprim objects, we used the iMetabot model described in this article to design and optimize a set of fuzzy controllers which were responsible for their final behaviour, which basically consisted of moving around the aquatic environment while avoiding any obstacles they encountered on their way, displaying characteristic behaviour in terms of their way of moving and their manner of fleeing from predators. All of the controllers designed were optimized in a VTS which was specially designed for the purpose, using an evolutionary learning scheme such as the one described in this article.

The fitness function of each iMetabot was directly proportional to the distance travelled per unit of time during its lifetime, and was inversely proportional to the number of collisions, also calculated per unit of time. In this case we used larger populations as we were not limited by the number of simple objects present on a single island, nor did we need to design complex forms for objects as we were not taking animations into consideration (basic shapes are sufficient as long as we make sure that the forces and impulses applied are based on the mass of the objects used). Reproduction was a process which selected two metabots in proportion to the value of their fitness function. Genotype arrays were obtained for the progenitors, and two new genotypes were generated through multi-point crossover. All of this involved a mutation percentage of 0.5%, a reproduction percentage of 65% and an elitism percentage of 1%.

Figure 5: *Images from the UPM 2008 Competition Call.*

AFKBS model adaptation to this virtual environment has enabled us to build virtual objects with complex behaviors as the predators and preys of the competition. Likewise, our research determined that AFKBS perfectly fit virtual environments due to: (i) are easy to define since the rules of a FKBS are expressed in a language close to humans and (ii) simulated environments allow FKBS adjustment using evolutionary techniques. From 2008 to date we have experimented with this technique in many drivers and they all achieve their goal because their adjustment occurs in the same environment in which they play their behavior.

5.2. BabyBot project

As we have already mentioned, eMetabots are very different from iMetabots as the avatars only have three types of movement (walk, run and fly) and it is not possible to control their movements through the controlled application of forces and impulses. In this case, experimentation was carried out in the BabyBot project, the goal of which was to design a virtual child by Artificial Intelligence techniques for the detection of criminal behaviour in virtual environments. The main function of this virtual child is its conversational capacity, although it also needs to use mobility in order to be more credible. As previously mentioned, in other articles (Griol *et al.*, 2009; Griol *et al.*, 2010a, 2010b) we have described our experiments on the conversational capacity of metabots. For this project we have developed the high-level social functions to locate the objective points in which to place the BabyBot for each type of interaction, meaning that we have been able to increase the length of speech achieved by the eMetabot.

In this article the focus has been on avatar mobility, and we studied how this mobility can make our virtual child more popular among the avatars with which it converses. In addition to this, we have developed controllers such as the one shown in point 4.2 which allow us to move our eMetabot towards its objective without colliding with any obstacles (moving or stationary) that it meets on its way.

The results obtained with this movement control system along with those achieved in the conversational aspect, allowed us to build a virtual child simulating a large number of behaviours of a real child allowing Optenet to offer a commercial product for children protection on social networks and virtual worlds.

6. Discussion and future works

In SL, we are limited by the restrictions imposed by its servers. Having more than 100 avatars on a single island increases lag and makes experimentation impossible. As a result, during each of the training sessions carried out with avatars, we limited the number of eMetabots to 60 and, when we use iMetabots, we have to keep within the limitations imposed by the LSL which, in some aspects, are very restrictive (e.g., we only have a few KBs of memory available per script). This limitation forces us to carry out controller optimization processes outside the SL environment, converting the results for the controller into a table of pre-calculated values which are used by the iMetabots in their stage of obtaining fitness in the VTS.

The first clear conclusion obtained from the experiments presented is that metabots are a very promising field of experimentation for the implementation of the models used in Artificial Intelligence and for the development of human–machine communication in all its aspects and without the need for any specific learning by humans. An example of this is the promising results that we obtained: the increased human interaction with eMetabots and the substantial improvement of metabot mobility control (achieved with both types of metabot) through Adaptive FKBS (the number of collisions was considerably reduced, achieving optimal near-zero collisions during the entire lifespan).

There are just a few studies in the development of autonomous avatars and virtual environments, and even less those focused on their mobility. In fact, we could not find any work in

which soft computing techniques are used within a virtual world. In 2008, Ullrich *et al.*, say: 'Until now, There Seems To Be only one, very Specialized solution for agent in Second Life', telling about the work of Friedman *et al.* (2007). However, these studies are not useful for us since we can not make a comparison with our results because they just focus on nonverbal communication through gestures and animations and mobility reduces to what they call: 'a very basic type of navigation has been implemented (walking in random directions until either an obstacle or an avatar is found)' (Ullrich *et al.*, 2008).

The results obtained in our experiments, contrasted with our previous experiments in mobile Robotics (Serradilla and Gómez-Allende, 1997), allow us to evaluate the advantage involved in experimentation in persistent virtual worlds which provide social tools to a large number of users, compared with the cost and complexity of experimenting in the real world (no need for batteries, engines, sensors, etc.). Also, in recent work on this field (mobile robotics), we can find the use of FKBS in a very similar way we have just described in this article but without counting with the possibility of evolutionary adjustments. For example, Obe and Dumitrache (2010) uses FKBS to follow walls and avoid obstacles and says: 'Comparative analysis of performance of the two controllers shows better performance of Gaussian membership function based fuzzy logic controller over the Triangular membership function based fuzzy logic controller'. In future investigations we will focus our efforts on applying and merging techniques and models which are typical of AI in the design and construction of metabots.

On the one hand, we are interested in following the path already started by Breazeal with his robot Kismet (Breazeal, 2003) which relates to the maintenance of face-to-face interaction between the human and the robot (head position, gaze direction and facial expressions). In our case we tackle the problem from the perspective of metabots developing in virtual environments, in comparison to the excellent work by Breazeal carried out from the point of view of robotics. We will also continue our research in the field of chatterbots through interlocutor modelling (language, slang, age, etc.) with respect to text-to-text dialogue. Owing to the ever-increasing use of the voice in metaverses, we also intend to broach speech-to-speech dialogue (using open code libraries which are currently available) for interlocutor modelling in the same was as in text-to-text, but also adding elements which are unique to this style of dialogue (prosody).

On the other hand, we will extend our current work in the field of semantic representation of behaviours in 3D persistent virtual worlds (Arroyo *et al.*, 2010), providing an introduction to knowledge representation and reasoning with ontologies and describing the spatial and temporal knowledge used by metabots. Connecting to a virtual world which is semantically isomorphic to reality, through an avatar, allows the concepts used by humans and those used by machines to move closer together. By defining a common ontology we can enhance the meaning of the communicative act. This experience storage and retrieval leads to experience sharing (Kim *et al.*, 2008) and the semantic representation of these experiences will be the core in many of the future developments in Mirror Worlds, Augmented Reality and Semantic Metaverses. Finally, we will persist in the development of our software bases MetabotLib and BotBridge, incorporating all of these new features as they are tested on our island in SL (http://slurl.com/secondlife/Tesis).

The continuity of our research has been achieved due to success with both examples presented in this article. Thanks to the success of 2008 UPM competition call, we have once again been granted funds to hold another competition (this time international) which we are working on at the moment. This new competition is called AvatarBall, and the objective is a virtual soccer tournament in which the groups of university students taking part will have to design their own soccer teams and compete against other teams. As in the previous contest, before the competition stage, we will equip virtual spaces (in this case soccer pitches) on our island in SL so that the teams taking part

can test their designs. Evidently, we are developing our own soccer team so that when competitors perform their tests they will have the chance to play against a fully operational team. And, thanks to the success of BabyBot project, we are working on a new project that enlarges the target to all kind of users with any potential vulnerability in virtual worlds and games.

Acknowledgements

This work has been partially supported by Optenet, project P08613023 'BABYBOT: Creating a 'virtual child' as a conversational agent, artificial intelligence techniques for the detection of criminal behaviour in virtual environments and social networking' (which was awarded the 4th Trofeo Red Seguridad (Security Network Trophy) in the category of 'Research into IT Security' for improvements in technology aimed at protecting young people on the Internet) and by the 2008 & 2010 UPM competition calls.

References

ARROYO, A., F. SERRADILLA and O. CALVO (2009) *Multimodal Agents in Second Life and the New Agents of Virtual 3D Environments. Methods and Models in Artificial and Natural Computation. Lecture Notes in Computer Science (LNCS)*, Vol. 5601, Berlin, Heidelberg: Springer, 506–516.

ARROYO, A., F. SERRADILLA and O. CALVO (2010) *Modeling Spatial-Temporal Context Information in Virtual Worlds. Methods and Models in Trends in Applied Intelligent Systems. Lecture Notes in Computer Science (LNCS)*, Vol. 6096, Heidelberg: Springer, 437–447.

BREAZEAL, C. (2003) *Designing Sociable Robots*, Cambridge: MIT Press.

CINGOLANI, P. (2005) jFuzzyLogic: Open Source Fuzzy Logic library and FCL language implementation. Registered in 2005. Last version: 2010. Available at http://jfuzzylogic.sourceforge.net/html/index.html (accessed 15 August 2010).

CORDÓN, O., F. HERRERA, F. HOFFMANN and L. MAGDALENA (2002) *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. Advances in Fuzzy Systems – Applications & Theory*, Singapore: World Scientific Publishing Company.

CHEN, Y., C. ZHANG and X. HUANG (2007) A Novel Vehicle Safety Traffic Pre-warning Decision on Multi-sensor Information Fusion. *Convergence Information Technology, 2007. International Conference*, 21–23 November 2007, pp. 318–322.

CHING-CHANG, W., H.-Y. WANG, S.-A. LI and C.-T. CHENG (2007) Fuzzy controller designed by GA for two-wheeled mobile robots, *International Journal of Fuzzy Systems*, **9**, 22–30.

CHRIETTE, A., T. CHEVIRON and F. PLESTAN (2009) Toward a general nonlinear model of reduced scale UAVs, *AIP Conference Proceedings*, **1107**, 154.

DICKERSON, J.A. and B. KOSKO (1993) Virtual worlds as fuzzy cognitive maps. *Virtual Reality Annual International Symposium*, 1993, 1993 IEEE, 18–22 September 1993, pp. 471–477.

Driankov, D. and A. Saffioti (Eds.) (2001) *Fuzzy Logic Techniques for Autonomous Vehicle Navigation. Physica-Verlag*, Germany: Springer-Verlag.

FRIEDMAN, D., A. STEED and M. SLATER (2007) *Spatial Social Behavior in Second Life*, Paris: Intelligent virtual agents. IVA.

GRIOL, D., A. ARROYO, M.A. PATRICIO and J.M. MOLINA (2009) Spoken Dialogue Systems for Enhanced Interactions in Virtual Worlds. *Third International Workshop on User-Centric Technologies and applications* (MADRINET'09). University of Salamanca (Spain). 10th–12th June 2009.

GRIOL, D., M.A. PATRICIO, J.M. MOLINA, A. ARROYO, Z. CALLEJAS and R. LÓPEZ-CÓZAR (2010a) Integración de los Sistemas de Diálogo para la Interacción en Redes Sociales. *Procesamiento del Lenguaje Natural*, Revista no. 44, abril de 2010.

GRIOL, D., E. ROJO, A. ARROYO, M.A. PATRICIO and J.M. MOLINA (2010b) A Conversational Academic Assistant for the Interaction in Virtual Worlds. *Forth International Workshop on User-Centric Technologies and applications. CONTEXT'10. International Symposium on Distributed Computing and Artificial Intelligence 2010*. Valencia (Spain). 7th–10th September 2010.

HERRERA, F. and M. LOZANO (2009) *Fuzzy Evolutionary Algorithms and Genetic Fuzzy Systems: A Positive Collaboration between Evolutionary Algorithms and Fuzzy Systems. Computational Intelligence. Intelligent Systems Reference Library*, Vol. 1, Berlin, Heidelberg: Springer, 83–130.

HO, H.F., Y.K. WONG and A.B. RAD (2008) Direct adaptive fuzzy control for a nonlinear helicopter system, *Aircraft Engineering and Aerospace Technology*, **80**, 124–128.

HOLVE, R., P. PROTZEL, J. BERNASCH and K. NAAB (1995) Adaptive Fuzzy Control for Driver Assistance in Car-Following. *Proceedings of the 3rd European Congress on Intelligent Techniques and Soft Computing* – EUFIT'95, Aachen, Germany, pp. 1149–1153, August 1995.

IIDA, K., T. MORI and T. YASUNO (2009) Fuzzy control for omni-directional vehicle with ball wheels, *IEEE Transactions on Electronics, Information and Systems*, **129**, 2019–2026.

KIM, I.J., S.C. AHN and H.G. KIM (2008) Experience Sharing in Tangible Web based on Lifelog. *Proceedings of Asiagraph*, 2008.

KIM, H.M., J. DICKERSON and B. KOSKO (1996) Fuzzy throttle and brake control for platoons of smart cars, *Fuzzy Sets and Systems*, **84**, 209–234.

KROESKE, J., A. GHANDAR, Z. MICHALEWICZ and F. NEUMANN (2008) *Learning Fuzzy Rules with Evolutionary Algorithms. An Analytic Approach. Parallel Problem Solving from Nature. Lecture Notes in Computer Science (LNCS)*, Vol. 5199, Berlin, Heidelberg: Springer, 1051–1060.

LUKE, S., L. PANAIT, G. BALAN, S. PAUS, Z. SKOLICKI, E. POPOVICI, K. SULLIVAN, J. HARRISON, J. BASSETT, R. HUBLEY, A. CHIRCOP, J. COMPTON, W. HADDON, S. DONNELLY, B. JAMIL and J. O'BEIRNE (2009) ECJ 19: A Java-based Evolutionary Computation Research System. Available at http://cs.gmu.edu/~eclab/projects/ecj/. Last version 2009 (accessed 15 August 2010).

NARANJO, J.E., C. GONZÁLEZ, R. GARCÍA and T. DE PEDRO (2008) Lane change fuzzy control in autonomous vehicles for the overtaking maneuver, *IEEE Transactions on Intelligent Transportation Systems*, **9**, 438–450.

OBE, O. and I. DUMITRACHE (2010) Fuzzy control of autonomous mobile robot, *U.P.B. Science Bulletin, Series C*, **72**, 173–186.

PARENTHOËN, M., P. REIGNIER and J. TISSEAU (2001) Fuzzy Systems 2001. *The 10th IEEE International Conference on Fuzzy Systems*. 2–5 December 2001, Melbourne, Australia, Vol. 1, pp. 252–255.

PATRICIO, M.A., D. MARAVALL, J. REJÓN and A. ARROYO (2005) *A Neurocalibration Model for Autonomous Vehicle Navigation. Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach. Lecture Notes in Computer Science (LNCS)*, Vol. 3562, Berlin, Heidelberg: Springer, 519–528.

PHAM, D.T., M.H. AWADALLA and E.E. ELDUKHRI (2007) Adaptive and cooperative mobile robots. Proceedings of the Institution of Mechanical Engineers, Part I: *Journal of Systems and Control Engineering*. London: Professional Engineering Publishing. Vol. 221. No. 3, 2007.

RODRÍGUEZ-CASTAÑO, A., G. HEREDIA and A. OLLERO (2000) Fuzzy path tracking and position estimation of autonomous vehicles using differential GPS, *Mathware & Soft Computing*, **VII**, 257–264.

ROSS, T. (2007) *Foundations of Fuzzy Control*, New York: Wiley.

SERRADILLA, F. and D.M. GÓMEZ-ALLENDE (1997) Cognitive Modeling for Navigation of Mobile Robots Using the Sensory Gradient Concept, in *Proceedings of the A Selection of Papers From the 6th international Workshop on Computer Aided Systems theory* (February 24–28, 1997). F. Pichler and R. Moreno-Díaz, Eds. *Lecture Notes In Computer Science*, vol. 1333. Springer-Verlag, London, 273–284.

STEPHENSON, N. (1992) *Snow Crash*, New York: Bantam Books.

SUGENO, M., H. WINSTON, I. HIRANO and S. KOTSU (1995) Intelligent Control of an Unmanned Helicopter Based on Fuzzy Logic. *Proceedings of American Helicopter Society*, 51st Annual Forum, Houston, Texas, pp. 791–803, May 1995.

ULLRICH, S., H. PRENDINGER and M. ISHIZUKA (2008) MPML3D: Agent Authoring Language for Virtual Worlds. *Advances in Computer Entertainment Technology*. Yokohama, Japan.

USERO, L., A. ARROYO and J. CALVO (2007) *Context Information for Understanding Forest Fire Using Evolutionary Computation. Nature Inspired Problem-Solving Methods in Knowledge Engineering. Lecture Notes in Computer Science (LNCS)*, Vol. 4528, Berlin, Heidelberg: Springer, 271–276.

VACHKOV, G. and T. FUKUDA (2001) Structured learning and decomposition of fuzzy models for robotic control applications, *Journal of Intelligent & Robotic Systems*, **32**, 1–21.

VINGE, V. (1981) *Binary Stars, Vol. 5, Nightflyers / True Names*. Toronto: Dell publishing.

The authors

Angel Arroyo

Angel Arroyo is a Professor of Applied Intelligent Systems Department at Technical University of Madrid (UPM), Spain. He holds the BS and MS degrees in Computer Engineering at UPM and he is currently researching for PhD degree on the spatial and temporal semantics in virtual worlds. His main field of scientific interest as well as teaching area is the development of the Intelligent Agents and Systems. Also, he has worked in projects involving topics of Computer Vision, Expert Systems, Fuzzy Logic and Evolutionary Computing.

Francisco Serradilla

Francisco Serradilla was born in Sevilla, Spain, in 1965. He received the BS and the PhD degrees in Computer Science from the UPM. Currently, he is a Lecturer with the Department of Applied Intelligent Systems, UPM, and Director of the UPM research group in Intelligent Agents and

Ubiquitous Computing. His main research fields are Intelligent Information Agents, including Software Robots, Search Engines and Collaborative Recommender Systems. He is working on the collaborative filtering kernel of web site http://www.filmaffinity.com, and developing ubiquitous tools to interact with this site using iPhone device.

Oscar Calvo

Oscar Calvo is currently a Professor of Applied Intelligent Systems Department at UPM. He received the BS degree in Computer Engineering at UPM in 2009. His research areas include Intelligent Agents, Ubiquitous Computing, Immersive 3D Environments, Metaverses and Metabots.