# UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League

Juan Cristóbal Zagal and Javier Ruiz-del-Solar

Department of Electrical Engineering, Universidad de Chile,
Av. Tupper 2007, 6513027 Santiago, Chile
{jzagal, jruizd}@ing.uchile.cl
http://www.robocup.cl

**Abstract.** UCHILSIM is a robotic simulator specially developed for the RoboCup four-legged league. It reproduces with high accuracy the dynamics of AIBO motions and its interactions with the objects in the game field. Their graphic representations within the game field also possess a high level of detail. The main design goal of the simulator is to become a platform for learning complex robotic behaviors which can be directly transferred to a real robot environment. UCHILSIM is able to adapt its parameters automatically, by comparing robot controller behaviors in reality and in simulations. So far, the effectiveness of UCHILSIM has been tested in some robot learning experiments which we briefly discuss hereinafter. We believe that the use of a highly realistic simulator might speed up the progress in the four legged league by allowing more people to participate in our challenge.

## 1 Introduction

A fully autonomous robot should be able to adapt itself to the changes in its operational environment, either by modifying its behaviors or by generating new ones. Learning and evolution are two ways of adaptation of living systems that are being widely explored in evolutionary robotics [5]. The basic idea is to allow robots to develop their behaviors by freely interacting with their environment. A fitness measure determines the degree in which some specific task has been accomplished during behavior execution. This measure is usually determined by the designer. One of the main factors to consider within this approach is the amount of experience that the robot is able to acquire from the environment.

The process of learning through experience is a time consuming task that requires, for real robots, testing a large amount of behaviors by means of real interactions. An alternative consists on simulating the interaction between the robot and the environment. Unfortunately, since simulation is usually not accurate, the acquired behaviors are not directly transferable to reality; this problem is usually referred to as the *reality gap*. There is a large list of experiments in the literature where simulators are used for generating simple robotic behaviors [5][6]. However, there are few examples of the generation of complex robotic behaviors in a simulation with successful transfers to reality.

Simulation can be achieved at different levels, for example: one can simulate the high-level processes of robot behaviors by simplifying the sensor and actuator responses. A more complete representation is obtained when considering the low-level interactions among sensors, actuators and environment. Nevertheless, this usually entails complex models of dynamics and sensor related physical processes. We believe that a fundamental requirement for generating low-level behaviors from simulations is to consider a complete representation which includes low-level physical interactions. Thus, high-level behaviors can be obtained from lower-level behaviors in a subsumption fashion.

Nowadays, generating representative simulators of the dynamics of the interactions of robots might not be an impossible task. Once achieved it allows for the easy generation of a variety of complex behaviors which otherwise would take a very extensive design period. However, we believe that generating a realistic simulator is not just a matter of modeling and design. In order to be fully realistic, the simulator must be adapted through real robot behavior execution as proposed in [12].

The RoboCup four-legged league offers a great challenge and opportunity for exploring low-level behavior acquisition. In this context we have decided to investigate how the use of a very realistic simulator might help the development of new behaviors. Although this league simulation warrants a good degree of attention, we identify a lack of accurate dynamic simulators. Aiming at solving this gap we present UCHILSIM, an accurate simulator in both the dynamic as well as the graphic aspects. The main design goal of the simulator is to become a platform for learning complex robotic behaviors by testing in a virtual environment the same controllers that operate in the real robot environment. UCHILSIM is able to learn its own parameters automatically, by comparing the robot controller behavior fitness values in reality and in simulations. We believe that the extensive use of this kind of tool might accelerate the generation of complex robotic behaviors within the RoboCup domain.

The remainder of this paper is ordered as follows. In section 2 some related work is presented. The UCHILSIM simulator is described in section 3. In section 4 some real learning experiments with AIBO robots using UCHILSIM are shown. Finally, in section 5 the conclusions and projections of this work are presented.

## 2 Related Work

So far three simulators have been reported for the RoboCup four-legged league, these are the ASURA team simulator [3], the ARAIBO team simulator [1], and the German Team Robot Simulator [7]. All these simulators consider a graphic representation of AIBO robots and soccer environment, but only the German Team Simulator considers as well the dynamics of objects at an elementary level of representation. Table 1 summarizes the main characteristics of these simulators. We consider that none of them represent with great accuracy both the graphic and the dynamic aspects of the environment. In this context there is a lot of work to be done in order to generate a simulator that accurately mimics the interaction of a robot in a real environment. We believe that it is possible to generate accurate simulators at a level in

which it is feasible to learn complex behaviors with the successful transfer of these behaviors to reality.

**Table 1.** Main characteristics of simulators that have been reported for the RoboCup four legged league.

| Name of Simulator | Presence of Dynamics | Level of Graphics | Functionalities |
|---|---|---|---|
| ASURA Simulator | No dynamics. | Good graphic representations | Allows capturing AIBO image. |
| ARAIBO Simulator | No dynamics. | AIBO camera characteristics well treated. | Allows capturing AIBO image. |
| German Team Simulator | Present at an elementary level. | Elemental graphic representations. | Large set of functionalities, e.g. interfacing with monitor, calibration tools, etc. |

## 3 UCHILSIM

UCHILSIM is a robotic simulator designed for the RoboCup four-legged league. It is built on top of two processing engines: one in charge of reproducing the dynamics of articulated rigid bodies and the other in charge of generating accurate graphic representations of the objects and the soccer setting. The simulator also includes a variety of interfacing functions which allow the core system to be connected with all the UChile1 controlling modules and sub-systems, as well as to some learning mechanisms. The simulator includes a complete graphic user interface. Figure 1 shows a screenshot of the use of UCHILSIM.

The main design goal of UCHILSIM is to allow robots to acquire behaviors learnt in a representative virtual environment. The long-term goal of the simulator is to allow the generation of complex behaviors for RoboCup team players by efficiently combining the acquisition of knowledge in reality as well as simulations [12].

Most of the rigid body dynamics in UCHILSIM are modeled and computed using the Open Dynamics Engine (ODE) library [9], which is an industrial quality library for simulating articulated rigid body dynamics in virtual environments. ODE is fast, flexible and robust; it allows the definition of a variety of advanced joints and contact points under friction. ODE solves equations of motion by means of a Lagrange multiplier velocity model. It uses a Coulomb contact and friction model which is solved by means of a Dantzig LCP solver method. In UCHILSIM special attention has been provided for the modeling of servo motors and ball.

The graphic representation of the objects in UCHILSIM is obtained by using the Open Graphic Library (OpenGL) [10]. The CAD model of the AIBO robots was generated starting from standard data provided by Sony [11]. The changes made to this model were the incorporation of player's red and blue jackets, the renewal of the robot

colors, and the modifications in the original model in order to achieve greater accuracy.

The following is a description of the UCHILSIM main components, such as the basic architecture of the system, the dynamic and graphic engines, the user interface, the learning capabilities of the system and the object loader.


**Fig. 1.** Illustration of the UCHILSIM software and its user interface. It is possible to observe the display of various viewing options.

### 3.1 Basic Architecture

Figure 2 illustrates the basic architecture of UCHILSIM. It is possible to observe how the core functions of the simulator are interfaced with a variety of applications. In the core of the simulator the dynamic engine closely cooperates with the graphic engine to generate a representation of each object. By means of a learning interface, a set of parameters is interchanged with a learning algorithm which runs independently of the simulator. These parameters are as a rule for either: defining the simulator variables or the robot controller variables which are being adapted during the learning process. The user interface allows changing several variables of the simulation, such as the way objects are being rendered as well as the external manipulation of objects within the game field. An application interface allows running the overall UChile1 package [8] on the simulation. We are currently working on an Open-r application interface which will allow the compilation of any open-r code under our simulator. We believe that such kind of tool will be quite relevant for the development of the league since it will allow, for example, to realistically simulate a game against any team in the league. Another recently incorporated component is the VRML Object Loader which allows to quickly incorporate new robot models into the simulated environment by following a fast and reliable procedure.

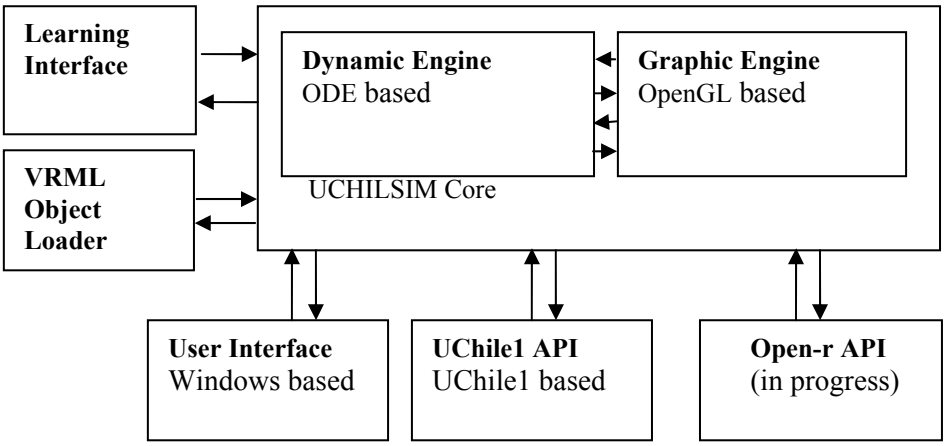


**Fig. 2.** Illustration of the UCHILSIM architecture. It is possible to observe how the overall system is organized as a set of interfaces around a core subsystem which contains the dynamic and graphic engines.

### 3.2 Dynamic Engine

In UCHILSIM all the AIBO body elements and the soccer field objects are modeled as articulated rigid bodies such as: parallelepipeds and spheres, which are connected to each other using different types of joints. A joint is a dynamic constraint enforced between two bodies, in order that they can only hold certain positions and orientations in relation to each other. We use universal joint models for defining the relationship among the AIBO torso and its thighs, i.e. rotation is constrained to just two degrees of freedom. Simple hinge joints are used for defining the relation among thighs and taps, i.e. constraining rotation to only one degree of freedom. Fixed joint models are used for defining the relation among the taps and hoofs; in this case one direction of deformation is allowed, and as a result the model accurately represents the rotation of the hoofs. In addition small spheres are attached to the base of each leg by means of fixed joints intended to mimic the effect of the leg tops.

The mass distribution of each rigid body is specified in the form of inertia tensors. For this implementation we assume a uniform distribution of mass for each rigid body. Mass estimation was carried out for each rigid body using a weight measuring device.

Collision detection is performed either with a simple model of spheres and parallelepipeds or by using a simplified version of the graphic grid which is attached to each rigid body. The latter approach is slightly time consuming although more accurate. We have obtained good results in all our experiments by using just the parallelepiped-sphere model. Figure 3 illustrate a diagram of the geometries which are alternatively attached to each rigid body for performing collision detection, it also shows the placement of the servomotors which are included in our model. They apply torque over joints according to the output of a PID controller which receives angular references given by the actuation module of the UChile1 software package [8].

On each simulation step the equation dynamics are integrated and a new state is computed for each rigid body (velocities, accelerations, angular speeds, etc). Then collision detection is carried out, and the resulting forces are applied to the corresponding bodies transmitting the effect of collisions along the entire body. Thus, the friction parameters deserve to be given special attention since they are used for computing the reaction forces between the robot limbs and the carpet. These parameters are under automatic adaptation on each performed experiment.

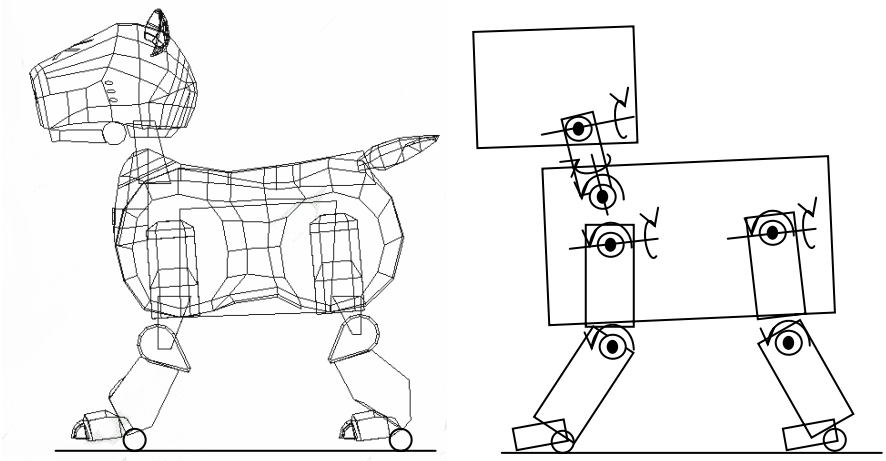

**Fig. 3.** Collision detection models which are alternatively used in UCHILSIM. The figure on the left corresponds to the graphic grid model used for collision detection. The figure on the right corresponds to the model generated with a set of parallelepipeds and spheres. By using this model we are able to perform accurate dynamic experiments while keeping the simulation at real time speed. The figure also shows the position of the servomotors which are included in the robot model.

### 3.3 Graphic Engine

The dynamic engine computes the corresponding positions and rotation matrixes of each body in the simulation space at the end of each simulation step. Then, for each rigid body, the corresponding graphic object is rendered. This is carried out by efficiently calling the corresponding graphic data. The graphic engine is also in charge of producing the image acquired from the AIBO's cameras. This is quite relevant for producing experiments with vision based systems. Using this system we will specifically intend to produce an extension of the work presented in [14]. We haven't concentrated our efforts on producing extremely realistic images yet, but we estimate that this process will be simple. We will incorporate some of the transformations which were proposed in [1] such as: the camera distortion and CMOS filters. Using a CAD modeler software we gave the AIBO models blue and red jackets, which were originally provided by Sony, we also constructed the corresponding soccer scenario. The process of importing the graphic data into C++ code was quite time consuming

before using the object loader. Currently the graphic data is directly obtained from the object loader module.

### 3.4 User Interface

The user interface of the simulator currently provides the following set of functions:

1. Loading of arbitrary AIBO models in modified VRML format.
2. Placement, at any moment, of different objects within the simulation, such as: robots, balls and other objects.
3. Arbitrary movement of objects while in motion, this is particularly useful while interactively generating games with the robots.
4. On line Modification of several parameters of the UChile1 controller.
5. Modification of several rendering options and viewing conditions, such as: wire frame representation, bounding box representation, point representation of objects, etc.
6. Management of the images captured by the AIBO's camera. These images can be exported into files or automatically transmitted to some learning software.
7. Loading and saving of a variety of configurations which define the game conditions.
8. Efficient management of several windows on the screen.

### 3.5 Learning interface

UCHILSIM is powered with a fast and efficient method for updating its parameters during running time. It is designed to communicate with other programs by means of a TCP/IP network. We have considered this, given that the simulator needs to adapt itself in order to perform experiments with the *Back to Reality* approach that will be discussed ahead.

### 3.6 UChile1 and Open-r API

The entire UChile1 software package, which allows a team of fully autonomous AIBO robots to play soccer, can be compiled for UCHILSIM. We had to carry out several modifications on our code in order to make this possible. However, the simulator is a great tool given that, besides its learning capabilities, it is very useful for debugging any piece of code of our system. We are currently working towards generating an Open-r API for the simulator; the idea is to be capable of compiling any Open-r code for UCHILSIM. We believe that there are several applications for such kind of tool, for example, it might speed up the progress of the league by allowing people around the world to develop and test software without the need of having a real AIBO. Thus, incoming research groups might collaborate with the league by testing

their code in a simulated environment. For those who already have a real AIBO it might be interesting to test their systems during long evaluation periods, letting teams play against each other during days. This will allow the accurate analysis of the differences among teams and of course the possibility of learning from these experiences.

### 3.7 Object Loader

The VRML language allows defining objects into a tree like structure of nodes, each one containing graphic as well as structural information of body elements, such as: mass, articulation points, etc. Although graphic data of AIBO models available to the public does not currently contain dynamic information, we have modified them by incorporating mass and motor data into the VRML files. On earlier versions of our simulator the robot models were hard wired into the simulator and the process of incorporating new models was quite time consuming. On the other hand updating a VRML file is considerably less time consuming. Using this technique we have added the ERS-220 and the new ERS-7 AIBO robot models into our simulator.

## 4 Using UCHILSIM for learning behaviors with Back to Reality

UCHILSIM is a platform intended for learning complex low level behaviors. The capabilities of UCHILSIM will be illustrated on hands of two different real experiments. We will first briefly describe *Back to Reality*. The Back to Reality paradigm combines into a single framework: learning from reality and learning from simulations. The main idea is that the robot and its simulator co-evolve in an integrated fashion as it can be seen in the block diagram presented on figure 4. The robot learns by alternating real experiences and virtual (in simulator) ones. The evolution of the simulator continuously narrows the differences among simulation and reality (reality–gap). The simulator learns from the implementation of the robots behavior in the real environment, and by comparing the performance of the robot in reality and in the simulator.

The internal parameters of the simulator are adapted using performance variation measures. When the robot learns in reality, the robot controller is structurally coupled to the environment, whereas when it learns in simulation the robot controller is structurally coupled to its simulator. Thus, the simulation parameters are continuously tuned narrowing the reality-gap along the behavior adaptation process.

The *Back to Reality* approach consists of the online execution of three sequential learning processes: *L1* is the learning of the robot controller in the simulated environment. *L2* is the learning of the robot controller in the real environment. *L3* is the learning of the simulator parameters. In this process the gap among simulation and reality is narrowed by minimizing, after each run of *L3*, the difference between the obtained fitness in reality and in simulation.

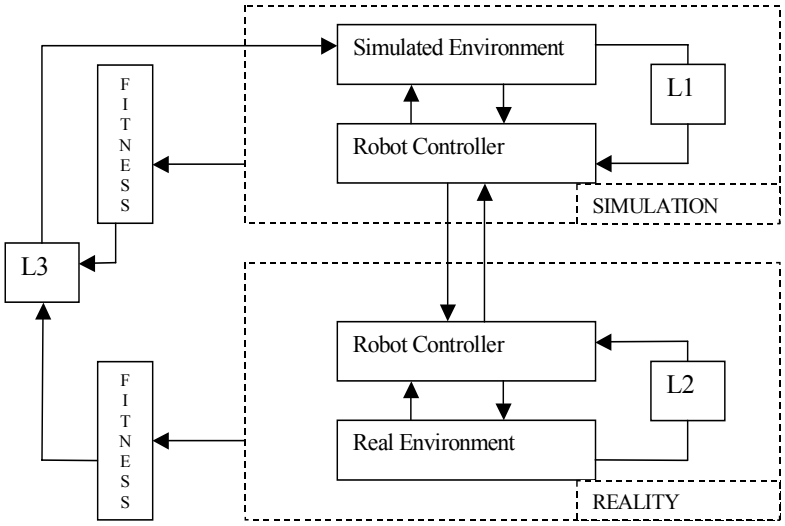


**Fig. 4.** Back to Reality building blocks.

During *L1* and *L2* the robot controller adaptation depends on the behavior *B'* observed in the simulated environment, and on the behavior *B* observed in the real environment, respectively. During *L3* the simulated environment is the result of the previous simulated environment and the real environment, as well as the real behavior and the simulated behavior. For implementing *L1* and *L2* any kind of learning algorithm can be used. Although we think that considering the evaluation time limitations of the experiments in reality, a reinforced learning algorithm is more suitable for implementing *L2*. Taking into account the flexibility of simulations, we think a good alternative for implementing *L1* are genetic algorithms. Regarding *L3*, we should consider the fact that the simulator has a large amount of parameters that probably are not explicitly related with aspects of the desired behavior. If this is the case, then *L3* could be implemented using genetic algorithms. Otherwise, reinforced learning could be an alternative. All these issues are addressed in [12].

### 4.1 Learning to walk

Since we have a team competing in RoboCup we are particularly motivated on improving the gait speed of our team. One can notice that there is a strong correlation between the speed of a robot-player and the success of a robot soccer team. We considerably improved the gaits of our system by learning with UCHILSIM and the Back to Reality approach. As a behavior fitness measure we used the robot speed measured in centimeters per second during evaluation trials of 20 seconds. The first stage of our experiment consisted on using genetic search for widely exploring a gait controller solution space. In this stage we use UCHILSIM as environment and a hand

tuned solution as a starting point (although our approach is well suited for a scratch starting point). The second stage consisted on evaluating in the real environment a set of successful behaviors, and measuring their corresponding fitness. These fitness measures are then used for optimizing the simulator parameters. The idea is that the simulator (UCHILSIM) be continuously updated. A genetic algorithm searches through the space of simulator parameters and minimizes the difference between fitness values obtained in simulation and their values obtained in reality. The third stage, which is simultaneously executed, consists on learning in reality using policy-gradient reinforcement learning. The idea is to take the solution that is obtained with genetic search under UCHILSIM, and then to perform smooth transitions around the solution by estimating the gradient, using the reinforcement method in the real environment. The best solution resulting from reinforcement learning is then transferred back to UCHILSIM where genetic search is again carried out. Then, the final stage consists on testing in reality the resulting solution by going *back to reality*. This process can be repeated indefinitely.

#### 4.1.1 Robot Controller Parameters

The following set of 20 parameters define the AIBO's gait in our experiments (for a detailed explanation see [12]): the locus shape (3 parameters: length, shape factor and lift factor.); the front locus shape modifier (3 parameters: lift height, air height and descending height); the rear locus shape modifier (3 parameters: lift height, air height and descending height); the front locus operation point (3 parameters: x, y and z); the rear locus operation point (3 parameters: x, y and z); locus skew multiplier in the *x*-*y* plane (for turning); the speed of the feet while in the ground; the fraction of time each foot spends on the air; the time spent on the lift stage (its equal to the descending time); the number of points in the air stage of which the inverse kinematics is calculated.

#### 4.1.2 UCHILSIM parameters

The robot simulator is defined by a set of 12 parameters, which determine the simulator and robot dynamics. These parameters include the ODE values used for solving the dynamic equations and the PID constants used for modeling the leg servos. There are 4 parameters for the mass distribution in the robot: head mass, neck mass, body mass and leg mass; 4 parameters of the dynamic model: friction constant, gravity constant, force dependent slip in friction direction 1 and force dependent slip in friction direction 2; and finally 4 parameters for the joint leg model: proportional, integral and differential constants of the PID controller and maximum joint torque.

### 4.1.3 Experiments Description

The procedure consists on learning the 20 robot controller parameters in the simulator and in reality, as well as learning the 12 simulator parameters. Genetic algorithms were used for the evolution of the simulator parameters and for the evolution of the robot controller parameters in UCHILSIM. Specifically, a conventional genetic algorithm employing fitness-proportionate selection with linear scaling, no-elitism scheme, two-points crossover with $Pc$=0.7 and mutation with $Pm$=0.005 per bit was employed. Given the set of parameters obtained from the simulator, we continued their adaptation in reality using the Policy Gradient Reinforcement Learning method [4]. The experimental conditions are fully described in [12].

First, walking experiments were carried out in UCHILSIM. The fitness evolution of these individuals is shown on figure 5. From these experiments we extracted a set of the 10 best individuals; they averaged a speed of 18 cm/s. The best individual of this group performed 20 cm/s in reality. The fitness value of each one of these individuals was compared with the resulting fitness that they exhibit in simulation. And the norm of the resulting fitness differences was used as a fitness function to be minimized by genetic search trough the space of simulator parameters. We obtained a minimum fitness of 2 cm/s as discrepancies occurred between the simulation and reality exhibited by these individuals. The best individual was then taken as a starting point for a policy gradient learning process performed in reality. With this method we achieved a speed of 23.5 cm/s. The resulting best individual obtained with reinforcement learning was then taken back to the now evolved adapted simulator where a genetic search took place starting with the population generated with permutations of this best individual. Finally some of the best individuals resulting from the genetic adaptation were tested on reality. Among these trials we found an individual that averaged a speed of 24.7 cm/s in reality. It should be noticed that improvements where done on our own controlling system, and that therefore, the resulting speed is not directly comparable to those obtained by others. Besides the gait locus used by a controller the efficiency on the computations also matter, the low level refinements on motor control, the inverse kinematics models being used, etc.

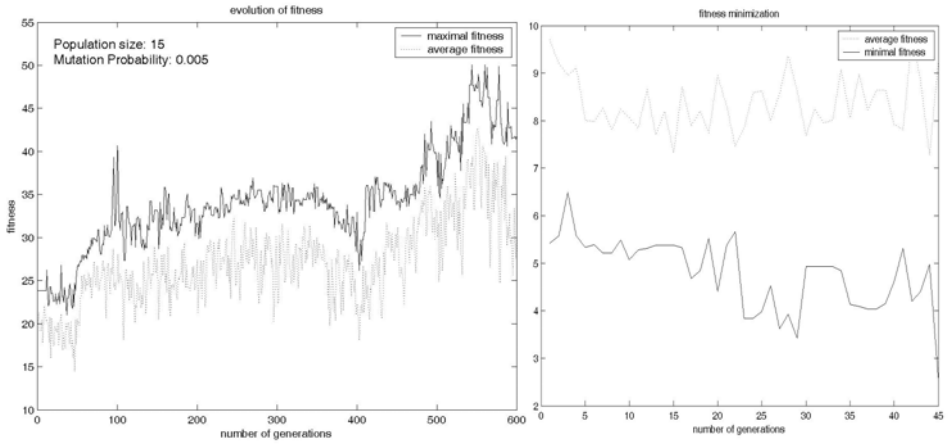


**Fig. 5.** Left: evolution of fitness for individuals tested with the UCHILSIM simulator in the first stage. Before adapting the simulator the individuals receive larger fitness in simulation than in reality. Right: Adaptation of the simulator, it is possible to observe how the minimization of the average differences of fitness obtained in reality versus simulations takes place.

### 4.2 Learning to kick the ball

Since this experiment is presented in [13] we will not offer many details here, however we can say that UCHILSIM was used for learning to kick the ball with AIBO robots using the Back to Reality approach, and that the resulting ball-kick behaviors were quite interesting, performing similarly as the best ball-kicks currently being used in the league. The behaviors which were obtained in the simulator were directly transferable into reality at the end of the adaptation process. Another important aspect to observe is that these behaviors were obtained from scratch.

## 5 Conclusions and Projections

UCHILSIM is a robotic simulator specially developed for the RoboCup four legged league. It reproduces the dynamics of AIBO motions and its interactions with objects in the game field with a great level of realism. The simulator also has a great amount of detail in the graphic representation of game field objects. The main design goal of the simulator is to become a platform for learning complex robotic behaviors by allowing testing of the same controllers that should operate in the real robot environment within a virtual environment. So far the effectiveness of UCHILSIM has been tested in two robotic behavior learning experiments which we have briefly described.

Currently each simulation step takes about 8*ms* with one AIBO robot being simulated on a Pentium IV 2.5 GHz processor and 512Mb of RAM. Using this

computing power we are able to simulate up to two robots at a realistic level. Several computers can be used for running simulations with more than two robots, however we have not implemented this option yet. We are currently working on improving our system; we expect to increase the current frame fate.

The following are some short term projections of UCHILSIM: (1) Consolidate the Open-r Universal UCHILSIM API. (2) Perform experiments where our localization methods will be tested using simulation and reality. (3) Perform experiments on the visual calibration of the simulator virtual cameras with basis on the Back to Reality approach. And (4) perform experiments combining real robots with virtual ones in a single soccer game.

A sample version of UCHILSIM is available at http://www.robocup.cl

## References


1. Asanuma, K., Umeda, K., Ueda, R., Arai, T.: Development of a Simulator of Environment and Measurement for Autonomous Mobile Robots Considering Camera Characteristics. Proc. of Robot Soccer World Cup VII, Springer (2003).
2. Google Source Directory Resource of Robotics Simulation Tools http://directory.google.com/Top/Computers/Robotics/Software/Simulation/ (2004).
3. Ishimura, T., Kato, T., Oda, K., Ohashi, T.: An Open Robot Simulator Environment. Proc. of Robot Soccer World Cup VII, Proc. of Robot Soccer World Cup VII, Springer (2003).
4. Kohl, N. and Stone, P. (2004). Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion. Submitted to ICRA (2004).
5. Nolfi, S., Floreano, D.: Evolutionary Robotics – The Biology, Intelligence, and Technology of Self-Organizing Machines, In: Intelligent Robotics and Automation Agents. MIT Press (2000).
6. Nolfi, S.: Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration. In: Gomi, T. (eds.): Evolutionary Robotics: From Intelligent Robots to Artificial Life, Ontario, Canada, AAI Books (1997).
7. Roefer, T.: German Team RoboCup 2003 Technical Report, Available at http://www.germanteam.de (2003).
8. Ruiz-del-Solar, J., Zagal, J.C., Guerrero, P., Vallejos, P., Middleton, C., Olivares, X.: UChile1 Team Description Paper, In: Proceedings of the 2003 RoboCup International Symposium, Springer, (2003).
9. Smith, Open Dynamics Engine Library, ODE web site available at http://opende.sourceforge.net (2003).
10. The Open Graphics Library. Available at http://www.opengl.org (2004).
11. The Open-r Software Development Kit. Available at http://www.openr.org (2003).
12. Zagal, J.C., Ruiz-del-Solar, J., Vallejos, P.: Back to Reality: Crossing the Reality Gap in Evolutionary Robotics. Proceedings of the IAV 2004, 5th IFAC Symposium on Intelligent Autonomous Vehicles, (in press), (2004).
13. Zagal, J.C., Ruiz-del-Solar, J.: Learning to Kick the Ball Using Back to Reality. Proceedings of RoboCup 2004: Robot Soccer World Cup VIII, Springer (in this volume), (2004).
14. Zagal, J.C., Ruiz-del-Solar, J., Guerrero, P., Palma, R.: Evolving Visual Object Recognition for Legged Robots, In: LNAI Proceedings of RoboCup 2003: Robot Soccer World Cup VII, Springer, (2003).