

Learning to Kick the Ball Using Back to Reality

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. Kicking the ball with high power, short reaction time and accuracy are fundamental requirements for any soccer player. Human players acquire these fine low-level sensory motor coordination abilities through extended training periods that might last for years. In RoboCup the problem has been addressed by engineering design and acceptable, probably sub-optimal, solutions have been found. To our knowledge the automatic development of these abilities has not been yet employed. Certainly no one is willing to damage a robot during an extended, and probably violent, evolutionary learning process in a real environment. In this work we present an approach for the automatic generation (from scratch) of ball-kick behaviors for legged robots. The approach relies on the use of UCHILSIM, a dynamically accurate simulator, and the *Back to Reality* paradigm to evolutionary robotics, a recently proposed method for narrowing the difference between simulation and reality during robot behavior execution. After eight hours of simulations successful ball-kick behaviors emerged, being directly transferable to the real robot.

1 Introduction

Kicking the ball with high power, short reaction time and accuracy are fundamental requirements for any soccer player. Until now all the four-legged teams use more or less the same method for kicking the ball, which works as follows: “First localize the robot with respect to the ball and targets, and then trigger some of the recorded ball kicks”. We consider that this approach is quite restricted since first, it requires the ball to be placed relative to the robot into a discrete set of positions, the problem with this is that the robot should invest valuable time on repositioning itself; and second, there is only a discrete set of target directions to chose among. Human soccer players are able to kick the ball freely, i.e. from almost whatever relative position and with any target direction. Moreover within the four legged league we can notice a direct relation among the amount of ball-kicking alternatives of players and the team success in robot soccer.

However, some basic requirements are needed before starting to think on applying non-constrained ball-kick behaviors. First robots should be able to estimate with accuracy the relative ball position at the right instant of ball impact, and second, robots should be able to arbitrarily control their limbs with precision. We consider that

these two requirements are currently fulfilled, at least by the best half portion of the teams competing in the league; therefore it is a good time for further exploring this approach. Even more, this ability will be of fundamental importance when playing soccer with humanoid robots. Hopefully, in a near future we will see very effective fast reactive behaviors, such as a robot in motion kicking the ball while it is also in motion or kicking the ball while it is on air, as human players do.

Another important aspect that we aim to address here is learning. Human players acquire these fine low-level sensory motor coordination abilities through extended training periods that might last for years. We believe that this type of reactive behavior can be derived from a tight interaction between the robot and the environment. There are several good examples in the literature of the successful generation of low level behaviors by learning from the real robot experience and practice [6][3][4]. However, learning to kick the ball with real legged robots seems to be an expensive and time-consuming task. We propose instead to use UCHILSIM, an accurate simulator, and the *Back to Reality* paradigm for combining virtual experiences with real ones. Our goal is to produce an optimal ball kick behavior for any given robot relative ball position and target direction that the robot might access without repositioning itself.

The remainder of this paper is organized as follows. In section 2 a short overview of RoboCup work related to this task is presented. In section 3 the general learning architecture that we have used is outlined. In section 4 the learning to kick the ball methodology is described. In section 5 the experiments are presented, and finally in section 6 the conclusions and projections of this work are presented.

2 Related Work

Kicking the ball is a key problem in RoboCup, however it has been addressed mainly at a high level within the RoboCup literature. Most of the work is concentrated on how to approach the ball and on how to choose the action to be triggered at the ball contact point. However some works are more related with low level behavior acquisition such as the work of Endo [1] in which a co-evolution of low level controller and morphology is proposed. The work of Hardt [2] surveys the role of motion dynamics in the design of control and stability of bipedal and quadrupeds robots. It is also related the work of Golubovic [5] which presents a software environment specially designed for gait generation and control design for AIBO robots. Certainly the work of Hornby [3] in the evolution of gaits for real legged robots is one important effort in order to produce low level behaviors from environmental interaction in a RoboCup directly related context. Beyond RoboCup the literature is quite rich in this subject, we recommend the reader to investigate further on the subject of evolutionary robotics [6].

3 The Learning Architecture

The experiments presented in this work are based on the recently proposed *Back to Reality* paradigm [7] to evolutionary robotics [6]. The main idea of this approach is to let the robots to behave both in their real environment and in a simulation of it. A fitness measure which tells the degree in which certain task is accomplished is computed after behavior execution. The difference in fitness obtained in simulation versus reality is minimized through a learning process in which the simulator is adapted to reality. In other words the robot continuously generates an internal representation of their environment while adapting the simulator. In this way, as figure 1 shows, the simulation parameters are continuously tuned narrowing the reality-gap along the behavior adaptation process. The adaptation of the simulator is dependent on the particular behavior that is executed, thus the behavior defines the way in which the world is sampled or observed. During adaptation three learning processes are taking place, first learning of the robot controller in the simulator, second learning of the robot controller in the reality, and finally learning of the simulator from real experiences. When the robot learns in reality, the controller is structurally coupled to the environment, while it learns in simulations the controller is structurally coupled with the simulator. Given these two couplings of the robot controller, it is necessary to conceive the robot as the union of the controller, the body and the simulator.

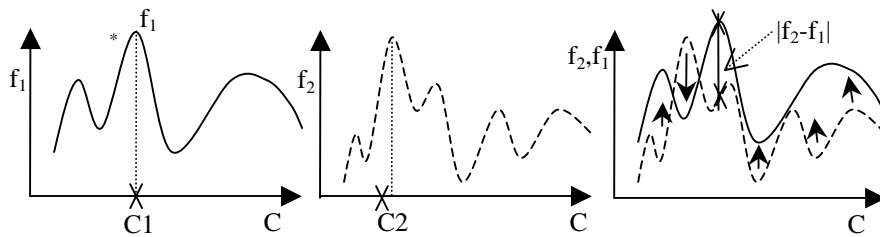


Fig. 1. How the reality-gap is narrowed by the Back to Reality paradigm (one iteration of the procedure is shown). The first/second curve shows the fitness curve f_1/f_2 over the controller parameter space in reality/simulation; the maximal value of fitness f_1^*/f_2^* corresponds to the C_1/C_2 controller. The last graph illustrates how the difference between both curves is narrowed during the evolutionary process; the fitness curve of the simulator, defined by the simulator parameters, converges towards the (real) fitness curve of reality, using the difference between the obtained real and virtual fitness as adaptation parameter.

We will use UCHILSIM [8], a dynamically accurate simulator for performing the adaptation of the controller under a virtual environment. We use genetic algorithms for searching ball-kicks solutions over the space of controller parameters. A fitness measure f_1/f_2 will be used in order to measure the degree in which each individual solves the proposed task. Genetic algorithms will also be used for evolving the simulator parameters such that $|f_2-f_1|$ is minimized during the learning process.

4 Learning to Kick the Ball

We aim at learning the best ball-kick behavior that the AIBO robot is able to execute while introducing a minimum of designer bias. As figure 2 (a) shows, the desired behavior should operate over a continuum of robot accessible ball positions and target directions. We define a ball-kick behavior domain as a continuous three dimensional space composed by the dimensions of the starting ball distance d [14cm \rightarrow 18cm] measured from the robots neck, the robot relative ball angle β [0 $^\circ$ \rightarrow 180 $^\circ$] measured from the robots right side and the ball target direction defined by an angle α [0 $^\circ$ \rightarrow 180 $^\circ$] measured from the robots right side.

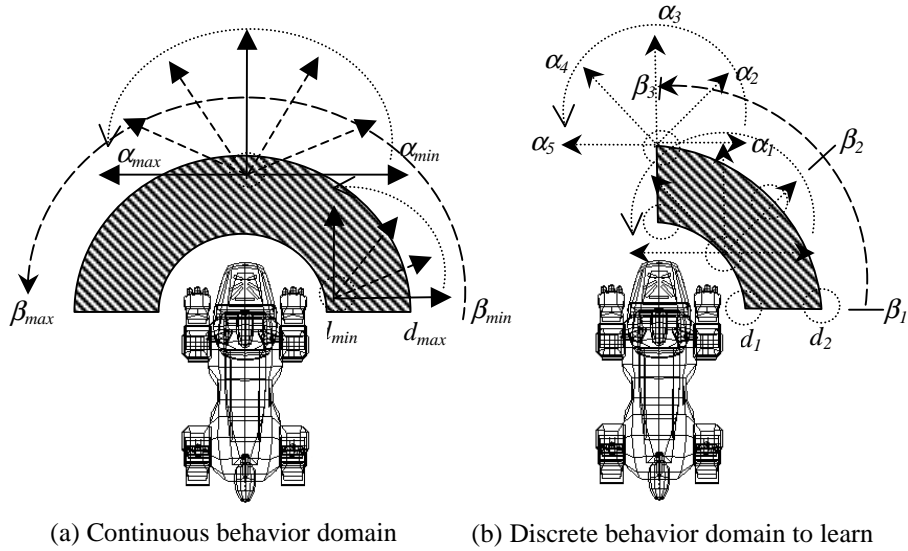


Fig. 2. Illustration of the behavior domain defined by different robot relative ball positions and ball target angles. They are first shown (a) for the expected behavior to be generated and (b) for the discrete set of points that we consider for learning.

In order to generate an optimal kick for each point (task) in this domain we will consider a discrete set of points for learning, as figure 2 (b) shows; two ball distances $\{d_1, d_2\}$, three relative angles $\{\beta_1, \beta_2, \beta_3\}$ and five target angles $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, i.e. thirty points in total. We have selected only the right half portion of the range of β since we expect solutions to be symmetric around $\beta = 90^\circ$. For each one of these thirty points, a specific ball-kick behavior will be generated by means of learning the set of parameters that define the corresponding sequence of robots limb displacements given rise to a ball-kick. The final continuous behavior for an arbitrary point, i.e. ball position $\{d, \beta\}$ and target angle $\{\alpha\}$, will be obtained by interpolating, in parameter space, the solutions which are obtained at the neighboring points. In all cases the idea

is to maximize the distance traveled by the ball while maintaining high accuracy in the resulting shooting direction.

4.1 Parameterization of ball kick

The way in which the ball-kick is parameterized might be a strong source of designer bias, therefore care must be taken in order to provide flexibility. We will present a ball-kick parameterization which aims at serving as a compromise among simplicity for learning and flexibility for allowing several solutions. Under a physical perspective we should consider that a larger amount of energy might be transmitted to the ball when all the robot limbs are allowed to move. For example, the legs supporting the body can displace the torso and its resulting ground relative speed might be aggregated to the torso-relative speed of the limb extreme which finally kicks the ball. It is also important to provide freedom with respect to where on the ball surface the forces are applied. This freedom might allow the robot to discover, just as an example, that kicking the ball on its upper part is more efficient for gaining speed than doing that on their lower part (as it happens in the billiard game). Notice that the UCHILSIM simulator well suites for such high detail level.

The configuration state of a limb is defined by the angular state of their three joints $\{\nu, \omega, \gamma\}$. The trajectory of a limb during a ball-kick might be described using hundreds of points in joint space, however we are analyzing only the case of fast reactive ball-kicks where no ball repositioning is required prior to ball-kick. For this case we estimate that it is sufficiently good to consider just four extreme points of displacements for each limb, which are (1) the starting limb configuration, (2) a limb configuration where energy is accumulated, (3) the limb configuration at the ball contact instant and (4) the ending limb configuration. In this case we consider the ending configuration to be the same as the starting limb configuration. Therefore for a given ball configuration and target angle a ball-kick correspond to the subsequent execution of the following four stages, which are depicted on figure 3:

1. The robot limbs are in their corresponding starting equilibrium configuration which we have selected to be the same as our walking equilibrium configuration.
2. A transition is taken from the equilibrium configuration of each limb to their corresponding energy accumulation configuration $\{\nu_{i1}, \omega_{i1}, \gamma_{i1}\}$, with $i=[1, \dots, 4]$.
3. Another transition is taken from the previous configuration to an energy liberation configuration (ball-kick instant configuration) $\{\nu_{i2}, \omega_{i2}, \gamma_{i2}\}$, with $i=[1, \dots, 4]$.
4. Finally the robot limbs are returned to their starting equilibrium configuration.

In this work 25 parameters should be obtained through learning, they correspond to six parameters $\{\nu_{i1}, \omega_{i1}, \gamma_{i1}, \nu_{i2}, \omega_{i2}, \gamma_{i2}\}$ for each one of the four robot legs, and one parameter which controls the speed of joints by incorporating certain amount of points

in the trajectory which are passed as intermediate references to the joint servo motors. A larger amount of interpolation points makes slower the motion of legs.

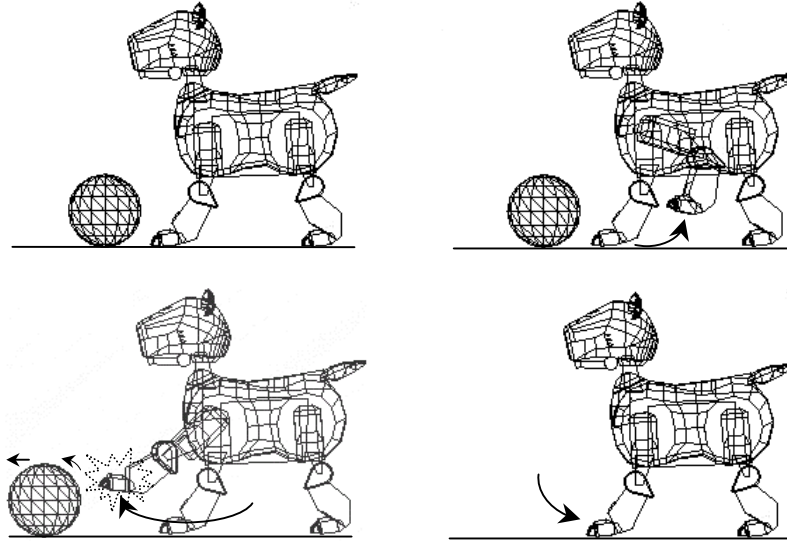


Fig. 3. Example of the subsequent configurations which are taken by one of the robot limbs during the execution of a ball-kick behavior. The limb starts from an equilibrium configuration (1), second it passes to a energy accumulation configuration (2), third it triggers the ball-kick (3) and finally the robot returns to the starting equilibrium configuration (4). Notice that for simplicity this illustration covers the action of just one limb, however the displacement of the four legs is considered in the presented experiments.

4.2 Simulator parameters

The robot simulator solves equations of motion derived from a Lagrange multiplier velocity based model [8]. We have selected a set of 12 parameters, which determine the simulator and robot model dynamics. These parameters include variables used for solving equations of rigid body dynamics and the PID controller constants used for modeling the leg joints. There are 4 parameters for the mass distribution in the robot: head mass, neck mass, torso mass and leg mass; 4 parameters of the dynamic model: friction constant, gravity constant, force dependent slip in two different directions; and finally 4 parameters for the joint leg model: proportional, integral and differential constants of the PID controller and maximum joint torque.

4.3 Genetic search

We use genetic algorithms for searching through the spaces of controller parameters and simulator parameters. In the case of the controller the search will be conducted in order to maximize the controller fitness function. In the case of the simulator the search will be conducted in order to minimize the difference of fitness obtained for a given behavior in simulation versus reality. We represent each parameter as a 8 bit string. Therefore we generate chromosomes of length 8x25 for the controller and 8x12 for the simulator. A range of variation is defined for each parameter p_i , such that $p_i \in [p_{i_{\min}}, p_{i_{\max}}]$. The algorithm used for evolution is a fitness-proportionate selection genetic algorithm with linear scaling, no elitism scheme, two-point crossover with a crossover probability $P_c=0.75$ and mutation with a mutation rate of $P_m=0.015$ per bit.

4.4 Definition of the controller fitness function

For a given ball position and target angle we aim at finding a ball-kick which maximizes the ball displacement while maintaining high accuracy in the resulting direction. Deriving a good fitness function for this purpose might appear quite straight forward, however some considerations must be taken.

Equation 1 corresponds to the first fitness function that we have used during our experiments, it corresponds to the measured Euclidean distance among the final ball position and the robot neck's position multiplied by the cosine of the error angle χ between the resulting direction vector and the target direction vector. This function is maximized when the error is zero.

$$fitness = d \cos(\psi) \quad (1)$$

However this function was shown to be useless for finding accurate ball-kick behaviors since genetic search first concentrates on finding individuals which are able to shoot the ball at large distances without having an accurate angle. The reason is that the cosine function is quite flat around zero. In order to solve this inconvenient we choose instead an exponential function as shown in equation 2. Using this function we observed that genetic search first concentrates on producing the right angle and then maximizes the distance.

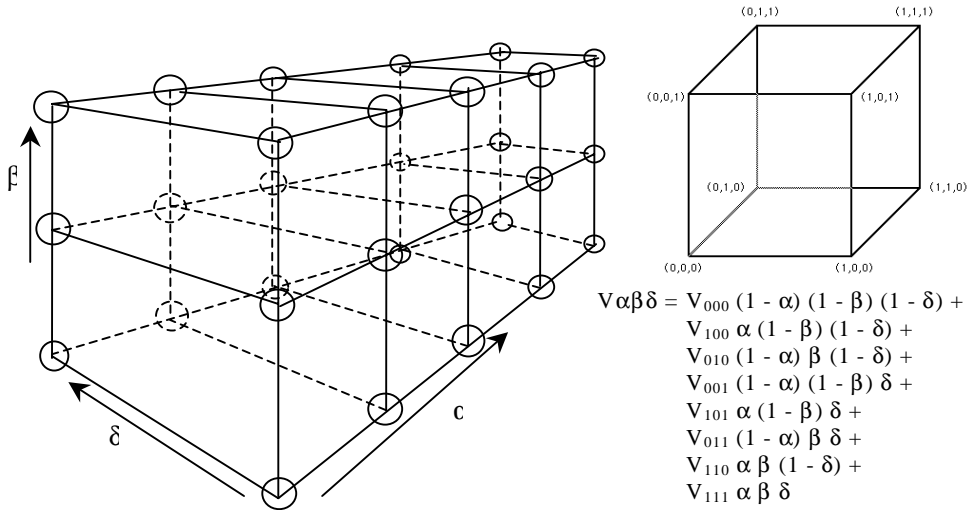
$$fitness = de^{-k\psi^2} \quad (2)$$

4.5 Combining solutions in parameter space

A different ball-kick behavior will be obtained for each one of the 30 points in the discrete behavior domain space. They will be obtained through genetic search over the

space of the 25 controller parameters using the back to reality approach. However the desired behavior should perform the right ball-kick for an arbitrary point $\{d, \beta, \alpha\}$ on the continuous behavior domain. For doing that we use trilinear interpolation over the obtained points. Figure 4 (a) shows an illustration of the three dimensional behavior domain grid which will be generated. Each corner in the grid contains the resulting 25 parameters of the best obtained behavior for the particular point. Figure 4 (b) shows the trilinear interpolation which is performed over each one of the 25 parameters in order to derive the corresponding interpolated value.

A requirement of this procedure in order to work is that neighboring grid solutions should produce a similar set of limb configurations. Otherwise the resulting interpolated ball-kick will not make sense. We measure this similarity as the Euclidean distance of the parameter vectors.



(a) Discrete behavior domain grid

Fig. 4. Illustration of (a) the three dimensional behavior domain grid, each point in the grid represent the set of 25 parameters which are obtained trough genetic search, and (b) trilinear interpolation which is performed for obtaining the 25 parameters for an arbitrary point $\{d, \beta, \alpha\}$ in the behavior domain.

5 Experiments

In this section we present the experiments which were performed with the methodology presented in 4. First we present some examples of the evolution of

fitness and resulting behaviors when learning a single isolated point in the behavior domain space. Finally we present different results on learning the whole discrete behavior domain grid and deriving a final overall continuous behavior.

5.1 Learning isolated points

For learning a particular isolated ball-kick behavior we first randomly generated an initial population of controllers according to 4.3. A known initial solution was used for defining the set of 12 simulator parameters (this solution was obtained from previous experiments on learning to walk [7]). Then the process continues as follows:

1. Each controller is tested on the simulator and fitness is computed according to equation 2. The evolutionary process runs along 50 generations and then it stops triggering an alarm to the user.
2. Human intervention is required in order to test a group of the resulting 10 best individuals in the real environment and to manually measure their corresponding fitness. Notice that this process can be automated, however it is not the main goal of this research.
3. Then the evolution of the simulator stage takes place in order to minimize the average differences among fitness values which were obtained in reality versus those obtained in simulation. For this each tested behavior is repeatedly executed using different simulator parameters derived from a genetic search process. This stage runs each time along 10 generations.
4. Once the simulator parameters have been updated the whole controller adaptation process continues going back to point 1. However the updates to the simulator are executed now every 100 generations instead of 50.

We ran this process for the 30 different points of the behavior domain grid that we aim at constructing. The above mentioned process was performed along 500 generations for each point. Figure 5 shows the evolution of fitness for the point $\{d=14cm, \beta=90^\circ, \alpha=90^\circ\}$, i.e. the task was to kick the ball when it is placed at 14cm right in front of the robot neck with a frontal target direction. The vertical lines represent the interruption stages where the simulator was adapted using as feedback the fitness measures of the resulting 10 best behaviors on the real environment. It can be seen how updates in the simulator produce differences in the maximal and average fitness along the controller adaptation process. We observed that at the beginning of the adaptation process the behaviors obtained in the simulator were not directly transferable into reality, i.e. their corresponding fitness was quite different, however at the end of the process (beyond generation 300) the resulting behaviors were almost directly transferable to reality.

Similar observations were done in most of the remaining 29 points of the behavior domain, however learning was not successfully accomplished in some points like $\{d=14cm, \beta=0^\circ, \alpha=180^\circ\}$. Certainly it is hard to imagine the robot performing a kick to the ball being placed at its right side with a left side target direction!. Some few neighbors to this point neither exhibit good results during the learning process. We consider that this is a design problem which can be easily solved by restricting the

behavior domain. For the presented experiment we replaced the neighboring good solutions in those points where solution was not found (just 4 cases).

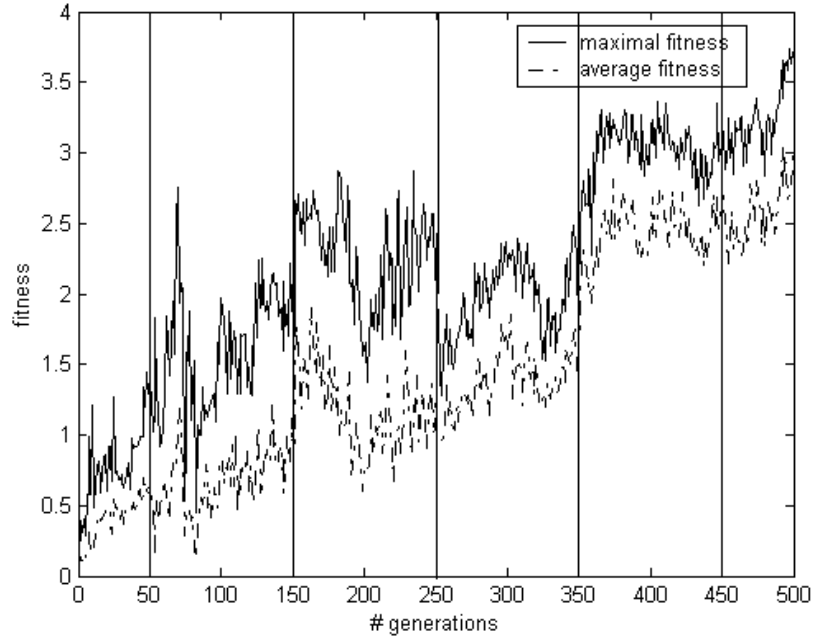


Fig. 5. Evolution of maximal and average fitness for the adaptation of the behavior domain point $\{d=14cm, \beta=90^\circ, \alpha=90^\circ\}$, i.e. the task was to kick the ball when it is placed at 14cm right in front of the robot neck with a frontal target direction. The vertical lines represent the interruption stages where the simulator is adapted using the feedback obtained when measuring the fitness of resulting best behaviors on the real environment. It can be seen how updates in the simulator produce differences in the maximal and average fitness along the controller adaptation process.

The resulting behaviors were quite interesting, even more if we consider that they were obtained from scratch. For example the robot learns that a way of shooting away the ball was to jump over it with its torso or with some legs as well. We obtained several behaviors where the concept of “accumulating energy” was heavily exploited. Some resulting ball-kick behaviors don’t look very nice since the robot ends up in strange configurations, however they are really effective. Figure 6 shows a screenshot of UCHILSIM while learning a ball-kick behavior.



Fig. 6. Illustration of using the UCHILSIM simulator for learning to kick the ball. It can be seen in this case a curious behavior where the robot first pushes its body forwards and then kicks the ball with its right leg.

5.2 Learning combined points

From the resulting grid which was constructed in 5.1 one can directly generate a continuous behavior for any point $\{d, \beta, \alpha\}$ using trilinear interpolation as it is described in 4.5. Unfortunately there is no guaranty of the success of points different from the grid points. This is due to the great variety of solutions which were obtained. For example two neighboring points in the grid producing a similar behavior, one sends the ball at direction 0° and the other at 45° , however one uses the left leg while the other uses the right leg. Interpolating these two behaviors gives rise to a behavior which let the robot to just fall over without even touching the ball. Under this scenario we visualize three alternatives: (1) to manually select similar solutions for the behavior grid disregarding at some point their optimality, (2) guide the evolutionary process by contaminating populations with manually selected individuals from neighboring points and (3) enforce similarity during the learning process by redefining the fitness function. We have produced a sufficiently good continuous overall behavior by using the second methodology. Starting from the point presented in 5.1 we contaminated their neighbor's populations with an 80% of individuals belonging to this successful point, and then we repeated the process over the complete domain. We really dislike this approach since it is quite time consuming and it goes against the philosophy of this learning approach, however it works well producing a continuous ball kick behavior over the entire domain.

6 Conclusions and Projections

It was presented a method for learning to kick the ball using virtual and real experience. The resulting behaviors are directly transferable from the simulator to the real robot. A few designer bias is introduced when (i) designing the ball-kick parameterization and (ii) when selecting some of the individuals for producing a continuum behavior. The resulting behavior allows kicking the ball with high power, short reaction time and accuracy over the entire domain. The employed method relies on the use of UCHILSIM, a dynamically realistic simulator, and the *Back to Reality* paradigm to evolutionary robotics. We remark that this method allowed us to learn from scratch a ball-kick which performs similarly as the most powerful ball-kick which is currently used in the league, i.e. when the robot leaves its body to fall over the ball (it was used for example by the German Team during the penalty definition against CMPack during RoboCup 2003). All independent ball kicks were obtained after just 8 hours of simulations. We should notice that the alternative is to let a student designing a ball-kick during a probably larger period of time. We are currently working on a method that automatically enforces similarity among behaviors in the grid. The fitness function contains a factor which depends on the Euclidean distance among the parameter vectors defining neighboring points. The key of this approach is to perform the search in parallel. That way allows us to maintain the 8 hours period of learning. Movies of some resulting behaviors, as well as a demo version of UCHILSIM are available at <http://www.robocup.cl>.

References

1. Endo, K., Yamasadki, F., Maeno, T., Kitano, H.: Co-evolution of Morphology and Controller for Biped Humanoid Robot. In: LNAI Proceedings of RoboCup 2002 Robot Soccer World Cup VI, Springer, pp. 327-339 (2002).
2. Hardt, M., Stryk, O.: The Role of Motion Dynamics in the Design, Control and Stability of Bipedal and Quadrupedal Robots. In: . In: LNAI Proceedings of RoboCup 2002 Robot Soccer World Cup VI, Springer, pp. 206-221 (2002).
3. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous Evolution of Gaits with the Sony Quadruped Robot. In: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp. 1297-1304 (1999).
4. Husband, P., Harvey, I.: Evolution Versus Design: Controlling Autonomous Robots. In: Integrating Perception, Planning and Action: Proceedings of the Third Annual Conference on Artificial Intelligence, Simulation and Planning, IEEE Press, pp. 139-146 (1992).
5. Golubovic, D., Hu, H.: An Interactive Software Environment for Gait Generation and Control Design of Sony Legged Robots. In: LNAI Proceedings of RoboCup 2002 Robot Soccer World Cup VI, Springer, pp. 279-287 (2002).

6. Nolfi, S., Floreano, D.: Evolutionary Robotics – The Biology, Intelligence, and Technology of Self-Organizing Machines, In: Intelligent Robotics and Automation Agents. MIT Press (2000).
7. 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).
8. Zagal, J.C., Ruiz-del-Solar, J.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. Proceedings of the RoboCup 2004: Robot Soccer World Cup VIII, Springer (in this volume), (2004).