Evolution of Controllers from a High-Level Simulator to a High DOF Robot

G. S. Hornby[1], S. Takamura[2], O. Hanagata[3], M. Fujita[3], and J. Pollack[1]

[1] Computer Science Dept., Brandeis University, Waltham, MA
{hornby, pollack}@cs.brandeis.edu
http://www.demo.cs.brandeis.edu

[2] ER Business Incubation Dept., Sony Corporation, 6-7-35, Kitashinagawa, Shinagawa-ku
Tokyo, 141-0001 JAPAN
takam@pdp.crl.sony.co.jp

[3] Group 1, D-21 Lab, Sony Corporation, 6-7-35, Kitashinagawa, Shinagawa-ku
Tokyo, 141-0001 JAPAN
{hana, fujita}@pdp.crl.sony.co.jp

Abstract. Building a simulator for a robot with many degrees of freedom and various sensors, such as Sony's AIBO[4], is a daunting task. Our implementation does not simulate raw sensor values or actuator commands, rather we model an intermediate software layer which passes processed sensor data to the controller and receives high-level control commands. This allows us to construct a simulator that runs at over 11000 times faster than real time. Using our simulator we evolve a ball-chasing behavior that successfully transfers to an actual AIBO.

1 Introduction

One area of evolutionary robotics is evolving controllers for robots. The two approaches are to evolve with actual robots or to evolve controllers in simulation for use with an actual robot. Real robots are always a better model than a simulator, yet they are limited to going at real time. Simulation also has the advantage of being more convenient to work with because of the ease with which performance can be monitored and the robot and world can be reset to perform multiple trials. Simulation's shortcoming is that because it is an imperfect model, controllers created for simulated robots often perform differently on actual robots. Nevertheless there have been several instances of controllers evolved in simulation that have transferred to real robots: locomotion for a hexapod [Gallagher & Beer, 1992] and [Gallagher et al., 1996]; obstacle avoidance with a Khepera [Michel, 1995]; and others.

One approach to developing a simulator is to create a simulator based on data taken from a real robot, [Miglino et al., 1995] and [Lund & Miglino, 1996]. Actual robot sensor readings are used to create lookup tables. In their experiments they evolved a neural control system for a Khepera to move around a

[4] AIBO is a registered trademark of Sony Corporation

simple environment while avoiding obstacles. Limitations of this methodology are that it lacks a way of developing an accurate model of the physical dynamics of the world, it is only useful for creating a partial model of the sensors, nor would it scale well to more complex sensors, such as a digital camera.

Another approach to constructing a simulator is that of minimal simulations [Jakobi, 1998]. This methodology identifies features of the environment that are easy to simulate and are necessary and sufficient for the robot to perform the desired behavior. These features are reliably simulated for each trial. Other aspects of the simulation are varied for each trial so as to be unreliable. Controllers evolve using only those features of the simulator which were reliably modeled and thus transfer successfully to an actual robot. Successful transference from simulators constructed using this method has been achieved for various tasks with various robots, such as T-maze navigation with a Khepera, locomotion and obstacle avoidance with a hexapod.

Projects using the simulators listed above connected the robot controllers directly to the robot's sensors and actuators. For AIBO, controllers sit on top of sensor processing and locomotion modules. Consequently our simulator does not model raw sensor values or motor commands, instead it models the processed sensory data and effects of high level movement commands. By modeling processed sensor input and high level motor commands our simulator was much easier to create and faster to execute, achieving a speedup of over 11000.

The main motivation in constructing this simulator is for evolving controllers for actual AIBOs. Controllers evolved in simulation should perform similarly on an actual AIBO. To evolve controllers that will transfer we include an error value for sensor values and locomotion similar to that used in minimal simulations [Jakobi, 1998]. Error values are determined randomly at the start of each trial and fixed for the entire trial – as opposed to adding random noise throughout a trial. In this way evolved controllers will be tolerant to sensors and actuators performing in a small range about the simulator setting, yet the controllers will not depend on noise. Using our simulator we evolve a ball-chasing behavior for AIBO which successfully transfers to a real AIBO.

The rest of the paper is organized as follows. In section 2 we describe AIBO and the simulator. Section 3 contains a description of the control architecture being evolved. In section 4 we describe our experiments and results. Finally, in section 5 we summarize our work.

2 Simulator

In figure 1 we show one way of visualizing a robot system. Layer a consists of the high level controller, which is what is evolved in these experiments. Layer b is an intermediate processing layer between the controller and layer c, the robot's sensors and actuators. Layer d is the environment in which the robot exists.

In evolutionary robotics, controllers are typically connected directly to a robot's sensors and actuators, with no layer b. Simulators developed for such systems must then model sensor values and actuator movements. Modeling these

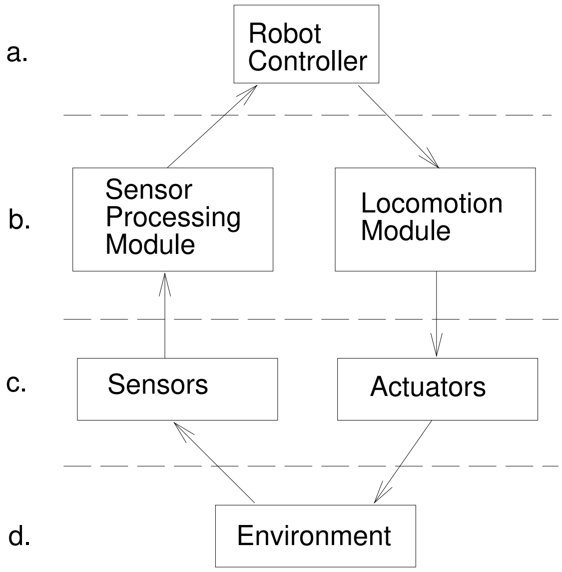


Fig. 1. Decomposition of a robot system.

parts of the system can require much development time and may require a high degree of accuracy, resulting in a slow simulation. Fortunately, AIBO comes with a layer b that processes sensor readings and converts high level locomotion commands to control the motors. Thus our simulator does not need to deal with various complex sensors and 14 motors, but need only model sensors in the way that the sensor processing module returns data – providing size and direction to each object instead of determining the color of each pixel of the digital camera – and it models the results of locomotion commands, such as *step forward* instead of modeling the movement of four legs, each with 3 DOF. On an SGI O2 with an R10000 processor our simulator performs approximately 27800 time-steps a second which, at 0.42s per simulated time-step, is a speedup of 11700 – adding processing for the neural controller reduced this to a speedup of 4500.

2.1 AIBO and its Environment

Our simulator models AIBO, a quadruped robot with 18 degrees-of-freedom and various sensors. We use 14 of AIBO's 18 degrees-of-freedom which are controlled through high-level commands, such as *look forward*, *look down* and which type of locomotion step to take. The sensing capabilities of AIBO that we utilize are the color detection tables of its digital camera, the infrared distance sensor, and joint values of the neck through various modules. Finally, the environment consists of a Robocup-like field and ball. A description of AIBO's prototype can be found in [Fujita & Kitano, 1998].

Walking and running is controlled through the locomotion module. This module takes high-level commands requesting different steps, *forward, front left, front right, turn left, turn right* and *backwards*, and moves AIBO appropriately. Gaits are developed by hand or are evolved [Hornby et al., 1999].

The two sensors we use are a distance sensor and a digital camera. The raw signal from the distance value is converted to a distance value in meters. The images from the digital camera are processed into 8 color detection tables, CDTs. For each table the total number of pixels containing the target color are given, as is the center of the pixel mass. One CDT is assigned to each object (own goal, opponent's goal, ball, teammate and opponent) in AIBO's environment – in these experiments only the CDT data for the ball is used.

The soccer field consists of a flat playing area bordered by walls. At each end a goal-area is marked by a section of colored cloth.

2.2 Simulator

Our simulator consists of a physical-dynamics component and a sensing component. The physical dynamics module handles movement of objects in the world as well as detecting and handling collisions between objects. The sensing component models the data sent by the sensor processing software to the robot's controller.

Fig. 2. Screen shot of the simulator, 1-on-1 with two AIBOs, two goals and a ball. Not shown are the walls lining the sides of the field.

Each AIBO robot is simulated as consisting of a body and a head. The different steps of the locomotion module are modeled as moving AIBO at a given linear and angular velocity. If AIBO's current linear or angular velocity are not the same as the specified value for the type of step being taken, acceleration is used to adjust these velocities according to classical dynamics. Similarly, AIBO's position is updated using classical dynamics.

At the end of each time step objects are checked for collisions. A collision occurs when two objects are found to intersect each other. When a ball collides with an object its velocity is instantaneously changed to its reflection. In addition, the following adjustments to ball velocity are used to model the effects of a ball colliding with a robot: if the ball collided with the front of an AIBO, the ball's velocity is increased by 10-80% of AIBO's linear velocity; if the ball collides with any of the other 3 sides of AIBO, the angle of its linear velocity is changed by up to 20% and its speed is reduced by up to 40%. When an AIBO collides with a wall it is moved perpendicularly away from the wall until it is just touching the wall. When two AIBOs collide with each other, they are moved away from each other along the line through their centers, figure 3, a distance such that they are no longer penetrating. In both types of collisions the linear velocity of the colliding robot(s) is reduced to 0.

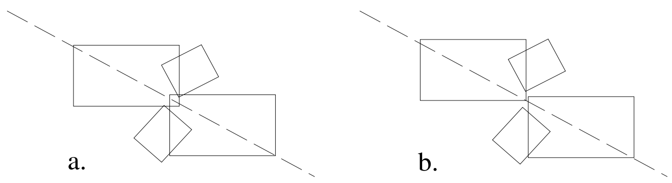


Fig. 3. Collision handling: when two robots intersect each other, the collision is handled by moving them apart along the line through their centers.

Modeling of sensing uses error to create uncertainty as with the methodology of minimal simulations. Before each trial the size of the error is decided at random for each sensor. The range of error is $\pm 10\%$ of the sensor value, a range in which the settings for an actual AIBO should lie. Once selected, this amount of error is used throughout the entire trial. A different error is selected for subsequent trials. This forces evolved controllers to perform robustly within the specified tolerance range, hence they should transfer to an actual AIBO. By not implementing the error as a random noise to be added throughout a trial, controllers do not evolve to become dependent on noise.

The most complex sensor to model is the digital camera, which is situated at the front of AIBO's head. A color detection table (CDT) is associated with each object – ball, own goal, opponent goal, teammate, opponent – in the world. For each color detection table, the CDT module returns the size, number of pixels of that color, as well as an and tilt angles to the center of the object visible.

Simulation of the CDT consists of calculating, for each object, the pan and tilt angles to the object's center and its distance from the digital camera. First the distance from the digital camera to the object is calculated. The object's leftmost and rightmost points are then calculated using the distance value along with the object's relative location and orientation from the camera. These values are clipped to be within the 52° field of view of the digital camera. Next, objects are compared for occlusion, with nearer objects occluding farther objects. The exception is that the ball does not obscure anything because of its small size. Once the left and right angles of all objects are known, the center is the average of these two angles. Object size, a measure of how large an object appears on the

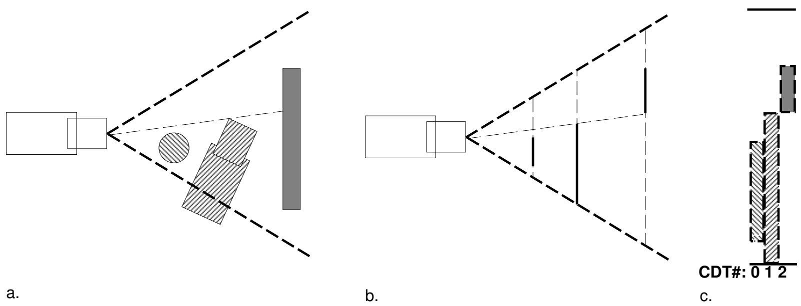


Fig. 4. Simulation of vision, showing the sequence of going from simulated world, to determining how large objects will appear on the robot's CDTs.

robot's digital camera, is calculated from the leftmost and rightmost boundaries on the CDT (see figure 4).

Once vision is completed the infrared distance sensor is handled. The distances from the front of head to the nearest wall, the floor, and any object within ($\pm 5°$) from a direct line of sight are compared. The smallest distance is the one returned as the distance value.

Modeling of both distance and vision sensing takes into account height of the digital camera and pan and tilt angles of the head.

3 Neural Control Architecture

The control architecture we use is a recurrent neural networks similar to that of [Hornby & Mirtich, 1999]. Neural networks consist of an input layer, a number of hidden units and an output layer. The neural controller receives input from AIBO's sensory module and sends commands to the locomotion module.

A neural controller consists of 12 input units, 20 hidden units and 7 output units. The data structure for a neural controller consists of: w, a matrix of real-valued weights; f, a vector of processing functions, $\{x, sin(x), asin(x), \frac{1}{1+e^{-x}}\}$; θ, a real-valued vector of input biases; and τ, a real-valued vector of time constants. The value of the ith row and jth column of the weight matrix is the value of the weight from the ith processing unit to the j processing unit. The ith value of the vectors is the attribute for the ith processing unit. A processing unit functions as follows:

$$a_{i,t} = \tau_i a_{i,t-1} + (1-\tau)[f(\sum_j w_{ji} a_j + \theta_i)] \quad (1)$$

As units are updated sequentially, the activation value a_j will be $a_{j,t}$ for values of j less than i and $a_{j,t-1}$ otherwise.

The twelve units of the input layer are listed in table 1. The first two units are the angles of the pan and tilt joints of AIBO's neck. The third unit is the distance from AIBO's distance sensor. The next five inputs correspond to stepping: forward; turn left; turn right; forward left; and forward right. The unit for

Unit	Value
0	tilt angle of neck (radians).
1	pan angle of neck (radians).
2	distance from infrared sensor (m).
3	last step was forward.
4	last step was turn left.
5	last step was turn right.
6	last step was front left.
7	last step was front right.
8	ball is visible.
9	size of ball in CDT.
10	tilt angle to ball.
11	pan angle to ball.

Table 1. Inputs to Neural Controller

whichever step was taken by AIBO in the previous time step has a value of 1, the other units have a value of 0. The remaining units are for visual sensing of the ball with AIBO's color detection table. There are units for: indicating whether or not the object is visible; object's size; pan angle to the object; and tilt angle to the object. If an object is not visible, then the reported size, pan angle and tilt angle are those for the last time the object was visible. Other inputs, such as vision information for the goals and other robots, are available but are not used in these experiments.

Unit	Value
0	stop
1	step forward
2	turn left
3	turn right
4	step front left
5	step front right
6	head position

Table 2. Outputs from Neural Controller

The 7 output units are listed in table 2. Outputs 0 through 5 are used to determine the locomotion behavior of AIBO. Whichever of these units has the highest activation value is the type of step that AIBO takes. Unit 6 determines the head position of AIBO. If the activation of this unit is greater than, or equal to, 0 then AIBO's head is moved to the *forward* position (-0.4 radians); otherwise the head is moved to the *down* position (-0.9 radians).

4 Experimental Results

To test our simulator we use it to evolve a neural controller for ball-chasing. Networks are recurrent neural networks with 12 inputs, 20 hidden units and 7 output units. and are evolved the same way as in [Hornby & Mirtich, 1999]. In these experiments we use a population size of 60 individuals, with 60% recombination, 40% mutation and an elitism of 2 individuals. Fitness is the average score over 12 trials – for each trial the score is a function of the distance which the ball is moved over a period of 21 (simulated) minutes.

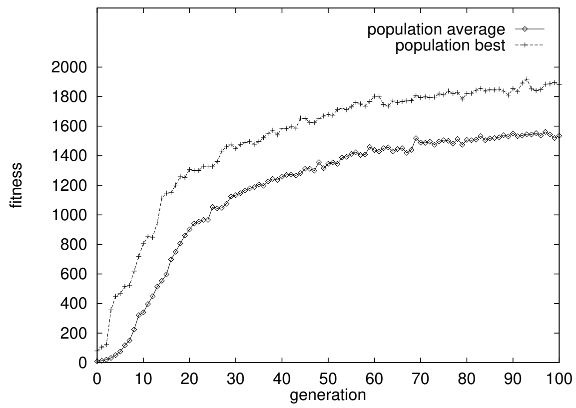
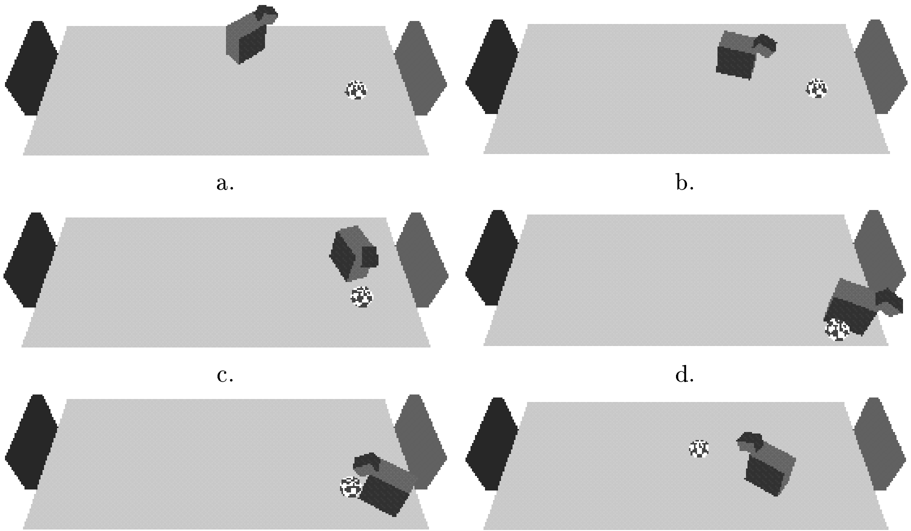


Fig. 5. Graph of average fitness for the entire population and fitness of the best individual, averaged over 6 evolutionary runs. Fitness is distance (cm) the ball is moved over a 21minute (3000 time step) trial.

Figure 5 contains a graph plotting the average fitness of 6 evolutionary runs. Individuals start with very poor ball chasing skills. After a few generations individuals are able to push the ball a few hundred centimeters over the course of 21 minutes. By the end of 100 generations individuals can push the ball more than 1500cm.

Evolved individuals chased the ball in a variety of ways. Initially they would turn until they could see the ball and move towards it. Some would push the ball a little, then re-center on it. Others would give a hard push and, once they lost sight of the ball, would then loop around until they spotted the ball again. All controllers performed the behavior well. Figure 6 contains screen shots of a chase sequence.

Individuals that were transferred to an actual AIBO were successful in chasing a ball. As in the simulated world they generally avoided the wall and were able to relocate the ball after it is moved to a different location in the world. Figure 7 shows pictures of an actual AIBO performing ball chasing with an evolved neural network controller.



a. b. c. d. e. f.

Fig. 6. Ball chasing sequence on the simulator.

One difference between simulation and the real world was in the collisions between the ball and the robot. In the simulator, collisions between the robot and the ball are treated like a collision between a ball and a moving box. In the real world, the robot has moving legs that cause a greater variety of results.

5 Summary

We described our simulator for AIBO. The simulator handled locomotion of the robot, sensing of the digital camera and infrared distance sensor, and physical dynamics between the objects in the world.

Using this simulator we evolved a neural-controller for ball chasing that successfully transferred to AIBO. We noticed that collisions between the robot and ball had different results in the real world than in the simulated world. This did not affect ball chasing performance but suggests that evolving more complex behaviors for AIBO to interact with a ball may require better modeling of robot-ball collisions.

References

[Fujita & Kitano, 1998] Fujita, M. & Kitano, H. (1998). Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robotics*, 5:1–14.

[Gallagher & Beer, 1992] Gallagher, J. C. & Beer, R. D. (1992). A qualitative dynamical analysis of evolved locomotion controllers. In Meyer, J.-A., Roitblat, H. L., & Wilson, S. W. (Eds.), *From Animals to Animats 2*, pp. 71–80.

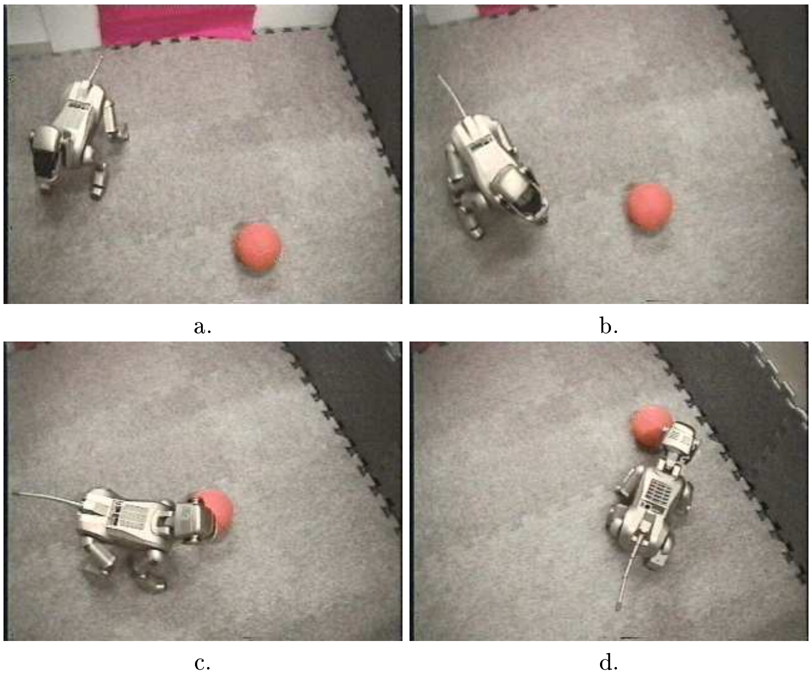


Fig. 7. Ball chasing sequence with an actual AIBO and an evolved neural network controller.

[Gallagher et al., 1996] Gallagher, J. C., Beer, R. D., Espenschied, K. S., & Quinn, R. D. (1996). Application of evolved locomotion controllers to a hexapod robot. *Robotics and Autonomous Systems*, 19(1):95–103.

[Hornby et al., 1999] Hornby, G. S., Fujita, M., Takamura, S., Yamamoto, T., & Hanagata, O. (1999). Autonomous evolution of gaits with the sony quadruped robot. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

[Hornby & Mirtich, 1999] Hornby, G. S. & Mirtich, B. (1999). Diffuse versus true coevolution in a physics-based world. In Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiel, & Smith (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

[Jakobi, 1998] Jakobi, N. (1998). *Minimal Simulations for Evolutionary Robotics*. PhD thesis, School of Cognitive and Computing Sciences, University of Sussex.

[Lund & Miglino, 1996] Lund, H. H. & Miglino, O. (1996). From simulated to real robots. In *Proceedings of IEEE 3rd International Conference on Evolutionary Computation*. IEEE Press.

[Michel, 1995] Michel, O. (1995). An artificial life approach for the synthesis of autonomous agents. In Alliot, J., Lutton, E., Ronald, E., Schoenauer, M., & Snyers, D. (Eds.), *Proceedings of the European Conference on Artificial Evolution*, pp. 220–231. Springer-Verlag.

[Miglino et al., 1995] Miglino, O., Lund, H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434.