Highly Modular Architecture for the General Control of Autonomous Robots

Ricardo A. Téllez, Cecilio Angulo and Diego E. Pardo

Technical University of Catalonia, Jordi Girona, 1-3,Barcelona, SPAIN

r_tellez@ouroboros.org, cecilio.angulo@upc.edu, diego.pardo@ben.colfuturo.org

Abstract The implementation in a robot of the coordination between different sensors and actuators in order to achieve a task requires a high formulation and modelisation effort, specially when the number of sensors/actuators and degrees of freedom available in the robot is huge. This paper introduces a highly distributed architecture that is independent from the robot platform, capable of the generation of such a coordination in an automatic way by using evolutionary methods. The architecture is completely neural network based and it allows the control of the whole robot for, in principle, any type of task based on sensory-motor coordination. The article shows how the proposed architecture is capable of controlling an Aibo robot for the performance of three different difficult tasks (standing, standing up and walking) using exactly the same neural distribution. It is also expected that it will be directly scalable for higher levels of control and general design in evolutionary robotics.

1 Introduction

Creating a control system for a quadruped robot like Aibo is a very challenging task, since the number of degrees of freedom available is high and the sensorial information to process is huge. To obtain a coordination between the elements in order to perform a global robot task is difficult and is a problem not completely solved yet in a general sense. Furthermore, the solution happens to be more difficult when the control mechanism is completely neural based and the way the different modules should interact is not programmed by hand. Some works have been developed that use completely neural based controllers on wheeled robots [1][2]. Those generate sensory-motor patterns required for the task, mainly using behaviour-based systems, but there are none applied to robots as complicated as a quadruped. An exception could be those works that address the problem of walking in quadruped or biped robots by using neural oscillators and Central Pattern Generators (CPG) [3][4][5][6], like for example the use of CPGs for the control of several postures and movements [7], but those are always focused on CPG dependent tasks (walking, running, scratching) and not having on mind a general purpose sensory-motor task.

Having as a goal the complete control of the robot by artificial neural networks, together with the idea of generic control (any robot, any task), this paper

introduces a general highly distributed architecture for the control of a robot on a sensory-motor coordination task. The idea is to have an architecture able to generate a coordination between different elements in a general way with independence of the task at hands and the robotic platform used. By taking as point of departure Minsky's idea of *society of mind* [8] and the idea of *modularity of mind* by Fodor [9], we see the robot's mind as a group of different modules each one in charge of its own device (sensor or actuator) that interacts with the rest of modules, where module is something similar to Fodor's definition: domain-specific processing systems, with their own proprietary transducers, and delivering non-conceptual outputs. The interaction of those modules produces as effect the accomplition of the task by the whole robot.

In order to show the results of our research, the article has been organised as follows. Section 2 describes the architecture developed including the learning mechanism. Section 3 shows the results obtained when the architecture was implemented on an Aibo robot in both simulation and real robot. Section 4 generates conclusions from the results and points towards future work based on those results.

2 Architecture Definition

The architecture is based on several uniform modules, composed of neural networks, where each module is in charge of one element of the robot. Through the use of a neuro-evolutionary algorithm, modules learn how to cooperate between them and how to control its associated element, allowing the whole robot to accomplish the task at hands.

2.1 Hardware

Using Fodor's definition of module, we define the Intelligent Hardware Unit (IHU) as a module created around a physical device (sensor or actuator). Every IHU is composed by a sensor or by an actuator and a micro-controller implementing an artificial neural network (ANN) that process the information of its associated device (received sensor information for sensors, commands sent to the actuator for actuators). It is said that the ANN is in charge of its sensor/actuator. This means that is the neural net the one that decides which commands must be sent to the actuator, or how must it be interpreted a value received from a sensor, and under which circumstances. All IHUs are interconnected to each other in order to be aware of what the other IHUs are doing. So in some sense, the net is also in charge of deciding what to say to the other elements as well as to interpret what the others are saying. The structure of a IHU can be seen in the following figure, together with a neural controller for a simple system with two sensors and two actuators.

It should be stated that when put several IHU together on a control task, each element has its own particular vision of the situation because each one is in charge of its own sensor or actuator. This leads to a situation where each unit

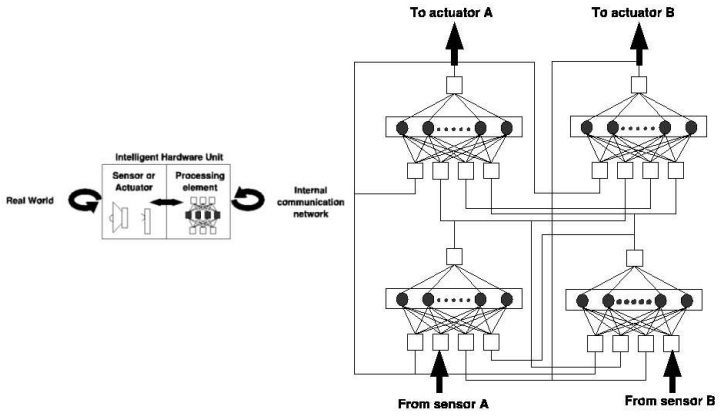

Figure 1. Schematics of an IHU, and the connection schema of four ANNs from four IHUs controlling a simple robot

knows what the others are doing but needs to select an action for its controller or sensor output, and based on its knowledge of the global situation and that of its particular device, decides what the next action will be.

Even though in the original definition a microprocessor was required for any IHU element, on the experiments presented here it has been simulated the existence of the micro-controllers linked to each device by allocating some processing time in the robot central processor for each IHU, since it was not physically possible to have one dedicated micro-controller for each IHU, neither in the simulations, nor in the real robot tests. It will be assumed that the results are not very different from the original idea.

2.2 Neuro-evolutionary algorithm

To teach the networks the coordination required for the task a neuro-evolutionary approach has been selected. For the Co-evolution of the different networks and due to the necessity of evolving different ANNs for different roles on a common task, a co-evolutionary algorithm is required. By using such kind of algorithm it is possible to teach to the networks how they must cooperate to achieve a common goal, when every network has its own an different vision of the whole system.

The algorithm selected to evolve the nets is the ESP (Enforced Sub-Populations) [10][11], which has been proved to produce good results on distributed controllers [12]. This algorithm is also in principle free of bias for a special task. It is a general algorithm which produces good results in systems where several parts must interact to obtain the general view of the situation

The chromosome codes in a direct way the weights of the network connections. A chromosome is generated for each IHU's network, and all are evolved at the same time over the same situation.

Usually in ESP, neurons in the hidden layer are also connected to themselves and to the other neurons of the same layer. This information is also coded in the chromosome. However, in the robot presented in this work and for the sake of simplicity, hidden neurons neither have been connected to themselves nor to the other neurons of the same layer.

3 Implementation

In order to validate the architecture explained in previous section, it was performed an implementation on a complex robot. Based on previous results where the architecture was tested on a very simple simulated robot [13], the architecture was later tested for the control of a complex real and simulated robot with several sensors and actuators while performing a task.

3.1 Validation with a complex robot

For a validation of the architecture on real robots the Aibo robot was selected. This is a complex robot with several degrees of freedom and multiple sensors and actuators that requires a good coordination between them to achieve any simple task. The aim for this stage was to see if the architecture was capable of controlling such a complex robot in the realization of some simple tasks and a more complicated one. First task was to keep an up position; second was to stand up from a lying position, and third task was to generate a gait behaviour.

For the realization of those experiments a simulator and a real robot were used. The simulator selected was the Webots program by Cyberbotics, which allowed the easy evolution of the controller without having to use the real robot [14]. This introduced easiness in the experiments since the simulator had the possibility of letting the system running alone doing training. Also damaging the robot and problems with batteries were avoided by using the simulator. Once the controller was generated and tested on the simulator, a direct connection was established to the real robot, that allowed the test of the resulting neural controller on it. At this stage of the Webots simulator development (version 4.0.28beta) only direct connection of the simulator to the real robot is available, since the *cross-compilation of controller for real Aibo* feature was not working yet for the ERS-7 model. Direct connection means that the controller runs on the computer (under Webots) but controls the real robot via wireless connection.

For the control of the robot the following sensors and actuators were taken into account:

- Actuators: four leg upper joints (J1), four leg middle joints (J2), four leg lower joints (J3). These are all motors that move Aibo's joints and determine its position on space.
- Sensors: four leg upper joints, four leg middle joints, four leg lower joints, four legs paw sensors. These are the sensors that indicate the state of the joint motors (actuators). Last four paw sensors indicate the state of the feet

Table 1. Parameters used during neuro-evolution

Parameter Name	Parameter Value
Subpopulations	8
Size of subpopulations	40
Mutation rate	0.4
Stagnation	20

paws. Those are switched on when the feet touches the ground or off when not touching. Furthermore, three accelerometer sensors (X,Y and Z) were used for determining Aibo's position.

This gives a total of 31 sensors and actuators, making the required number of ANNs to 31. All nets have the same number of inputs (31), outputs (1) and hidden units (8) for all the experiments. For the nets of the controllers, the inputs are connected to the outputs of all the other nets including itself. For the sensor networks are connected in the same way except that the entry from itself comes from the real sensor instead. For actuators, nets output indicate the kind of value that needs to be sent to the joint. For sensors, the output indicates the sensor value that is reported to the rest of IHU (including itself).

Values obtained from sensors were quantified, allowing a precision of one degree. This quantification generates a kind of dumping mechanism that prevents undesirable oscillations and never ending training. Network outputs of sensors were quantised in the same way. Network outputs of the actuators were quantized to provide only three possible values: move joint for 0.05 radians up, move joint for 0.05 radians down, or not move the joint.

Other required parameters for the evolution that were kept the same in all experiments are represented on table 1, and were selected based on previous experiences.

First test: keep standing up position The architecture was first tested to keep the robot on a standing up position . The robot was set up on an initial standing up position on the space and let free to act using its control networks. The goal was to keep up as high as possible with as less joint movements as possible. This seems a simple experiment but it is not because networks are directly connected to actuators and they keep on sending movement commands to the actuators all the time. This implies that joints will keep on moving unless the not-move-joint command is decided by their associated nets. This continuous movement of joints could lead to strange robot positions and eventually make it fall. In this task then, the robot must learn how to achieve a stable high position and keep it until the end of the evaluation time. For this purpose, the following fitness function was defined:

$$fitness = finalHeight/numberOfMovements$$

which evaluates the robot for its final height in opposition to the number of movement commands sent to the joints. Evaluation time for each phenotype was time steps.

After 13 generations the robot learned how not to fall dawn and how to keep a good structure to stay stable, but still too many movements were realized. After generation 28, movements were reduced to a a reasonable value while keeping the standing up position in both simulator and real robot.

Second test: stand up In this case the robot was required to change from a laying down position to a standing up one. For this purpose, the same architecture as in the previous experiment was used, been the only difference the fitness function and the time of evaluation of each phenotype, that in this case was of time steps.

The fitness function was defined as follows:

$$fitness = paws * finalHeight$$

Results showed that the robot was able to change its position to the desired one (see figure 2) after 77 generations in both simulator and real robot.

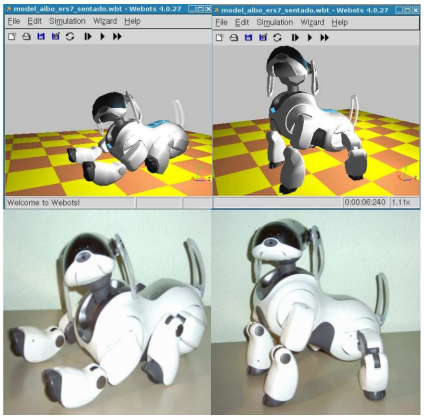

Figure 2. Figures showing initial laying down position and final stand up obtained position of Aibo in both simulation and real robot

Third test: walking This is by far the most difficult of the tests performed, and by hence, it was the one that gave more problems.

When designing controllers for robot walkers there are two main approaches: the first one uses a walking algorithm designed by hand by an engineer which indicates at any step the position of every joint of the robot [15][16][17]. Usually

this approach has a set of parameters to be tuned, and they can be done of several ways (by hand, by try and modify, by genetic algorithms, etc...). The resulting gait is one perfectly engineered that behaves exactly in the way the algorithm says.

The second approach consists of using CPGs. CPGs are non-linear oscillators made of neural nets. CPGs are coupled between them to generate the control signal for the actuators taking or not into account sensors information. The architecture developed here is very similar to that of CPGs, since it implements a group of neurons connected to the actuators, but it implements more connectivity and receives information from all sensors.

In this case, the architecture was implemented using recurrent neural nets instead of simple feedforward ones, and the fitness function was developed to conditionate the robot to acquire a determined gait style, based on the observance of Sony's walking style.

$$fitness = height * distance$$

Rule 1: fitness = 0 if J1 joints out of (-0.698,0.261) radians
Rule 2: fitness = 0 if J3 joints out of (0.349,1.658) radians
Rule 3: fitness = 0 if less than three paws down
Rule 4: fitness = 0 if after one front paw up does not follow a back paw up

For this experiment, an incremental evolutionary approach was taken, where the robot was let to neuro-evolve the controller for a few generations with rule 1 activated. After a determined number of evaluation steps rule 2 was activated, and so happens with rule 3 and 4. After applying four rules for several generations a basic walking pattern arise.

4 Discussion

Results presented here show that the neural architecture proposed is able to control a complex robot, and generate the required coordination mechanisms between sensors and actuators to perform complex tasks like walking or standing up. Coordination is learned during training phase and then used to solve the task during test phase with a completely neural based controller.

Experiments were performed in a general way, it is, there was no special consideration neither for the robot being controlled nor for the task being developed. Having implemented the architecture in such a way, it could be suggested that the architecture would be capable of the generation of any sensory-motor coordination required for any other task on any other robot, been the only limitations those imposed by evolutionary robotics. We admit thought, that the architecture has only been tested on two simulation robots (see author's previous work [13]) and only one real robot. Nevertheless, some research is been performed at present by the authors to apply this architecture to different types of robots on different tasks.

We think this work is a step closer to the final goal of obtaining a completely neurally controlled robot were more deliberative tasks could be performed.

ACKNOWLEDGEMENTS

Authors would like to thank Olivier Michel for their support and big help on Webots issues, and Auke Ijspeert for insightful suggestions and comments on walking robots.

References

1. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. The MIT Press (2000)
2. Floreano, D., Mondada, F.: Evolutionary neurocontrollers for autonomous mobile robots. Neural Networks **11** (1998) 1461–1478
3. Hiroshi Kimura, S.A., Sakurama, K.: Realization of dinamic walking and running of the quadruped using neural oscillator. Autonomous Robots **7** (1999) 247–258
4. Hiroshi Kimura, Y.F., Konaga, K.: Adaptive dynamic walking of a quadruped robot by using neural system model. Advanced Robot **15** (2001) 859–876
5. Collins, J., Richmond, S.: Hard-wired central pattern generators for quadrupedal locomotion. Biological Cybernetics **71** (1994) 375–385
6. Reeve, R.: Generating walking behaviours in legged robots. PhD thesis, University of Edinburgh (1999)
7. Billard, A., Ijspeert, A.J.: Biologically inspired neural controllers for a motor control in a quadruped robot. Proceedings of the International Joint Conference on Neural Network (2000)
8. Minsky, M.: The Society of Mind. Touchtone Books (1988)
9. Fodor, J.: The modularity of mind. Cambridge, MA: MIT Press (1983)
10. Gómez, F., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. In: Proceedings of the IJCAI99. (1999)
11. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Technical Report AI96-248, University of Texas (1996)
12. Yong, H., Miikkulainen, R.: Cooperative coevolution of multiagent systems. Technical Report AI01-287, Department of computer sciences, University of Texas (2001)
13. Téllez, R., Angulo, C.: Evolving cooperation of simple agents for the control of an autonomous robot. In: Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles. (2004)
14. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems **1** (2004) 39–42
15. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous evolution of gaits with the sony quadruped robot. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 1297—-1304
16. Hornby, G., Takamura, S., Hanagata, O., Fujita, M., Pollack, J.B.: Evolution of controllers from a high-level simulator to a high DOF robot. In: ICES. (2000) 80–89
17. Röfer, T.: Evolutionary gait-optimization using a fitness function based on propioception. Lecture Notes in Artificial Intelligence (2005)