# COLLABORATIVE CONTROL IN A HUMANOID DYNAMIC TASK

Diego Pardo and Cecilio Angulo
*GREC - Knowledge Engineering Research Group*
*UPC - Technical University of Catalonia*
*Avda. Victor Balaguer s/n. Vilanova i la Geltrú, Spain*

Keywords: Robot control architecture, Sensorimotor learning, Coordination policy, Reinforcement learning.

Abstract: This paper describes a collaborative control scheme that governs the dynamic behavior of an articulated mobile robot with several degrees of freedom (DOF) and redundancies. These types of robots need a high level of coordination between the motors performance to complete their motions. In the employed scheme, the actuators involved in a specific task share information, computing integrated control actions. The control functions are found using a stochastic reinforcement learning technique allowing the robot to automatically generate them based on experiences. This type of control is based on a modularization principle: complex overall behavior is the result of the interaction of individual simple components. Unlike the standard procedures, this approach is not meant to follow a trajectory generated by a planner, instead, the trajectory emerges as a consequence of the collaboration between joints movements while seeking the achievement of a goal. The learning of the sensorimotor coordination in a simulated humanoid is presented as a demonstration.

## 1 INTRODUCTION

Robots with several Degrees of Freedom (DOF) and redundant configurations are more frequently constructed; humanoids like Qrio (Kuroki et al., 2003), Asimo (Hirai et al., 1998) or HRP-2 (Kaneko et al., 2004), and entertainment robots like Aibo (Fujita and Kitano, 1998) are examples of it. Complex movements in complex robots are not easy to calculate, the participation of multiple joints and its synchronization requirements demands novel approaches that endow the robot with the ability of coordination. An attempt to drive the dynamics of its body optimally, measuring its performance in every possible configuration, would bring to a combinational explosion of its solution space.

The habitual use of simplified mathematical models to represent complex robotic systems, i.e., approximating non-linearities and uncertainties, would result in policies that execute approximately optimal control, thus, two major assumptions command the developed work. First, there is no pre-established mathematical model of the physics of the robot's body from which a control law could be computed; and second, the control design philosophy is focused on the action performance of the robot and not on the trajectory achievement by its joints; as an alternative, stochastic reinforcement learning techniques applied to numerical simulation models are studied. An optimal control problem, where a cost function is minimized to compute the policies, is stated.

The goal of this work is to solve a multi-joint robot motion problem where coordination is need. Planners and trajectory generators are usually in charge of complex motions where many joints are involved and the mechanical stabilization of the robot is in risk; the relationship between sensory signals and motor commands at the level of dynamics is viewed as a low level brick in the building of control hierarchy. Here we use a dynamic control scheme that also solves the trajectory problem.

It is important to mention that a previous work uses coordination at the level of dynamics to extend a manipulator robot capacity (Rosenstein and Barto, 2001), their objective was to use a biologically inspired approach to profit synergical movements between joints, like human muscles relationship. They use a hand-made pre-established PID controllers in

Pardo D. and Angulo C. (2007).
COLLABORATIVE CONTROL IN A HUMANOID DYNAMIC TASK.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 174-180
DOI: 10.5220/0001629001740180

every joint of the robot and the final applied torque in each motor is the linear combination of the PID's output, then collaboration is shown. The combination parameters are obtained by direct search methods (Rosenstein, 2003). By using linear controllers to solve a nonlinear problem, they implement a hierarchical motor program that runs various feedback controllers, i.e, they switch between PID's (parameters and goal) in the middle of the motion.

Here we extend that result by proposing a more general scheme, where sensor information is processed in independent and specific layers to produce coordinate control actions. Furthermore, at the architecture definition, the control functions are no restricted to PID's and its linear combination. Additionally, we use reinforcement learning to compute all the architecture controllers.

This paper is organized as follows Section 2 is devoted to the presentation of the architecture and a linear implementation of practical use is outlined in 3; while section 4 validates it through a simulation experiment of a robot equilibrium dynamical task. Section 6 gathers the conclusions and points to the future work.

## 2 LAYERED COOPERATIVE CONTROL

The Layered Cooperative Control (LCC) scheme is meant to generate motions while solving a *Dynamical Task* (DT). It is assumed that the final configuration of the DT is known, i.e. the set point of the joints angles is preestablished; but how to coordinate motions to reach this configuration, restricted to its body dynamics, is unknown. It is also assumed that the states of the system are observable and that each joint position is commanded by independent servos.

Figure 1 represents schematically the main idea of the LCC. The control actions that drive the motors are processed by two layers of controller functions. Each component of the first layer $(D)$ manipulates the dynamics of a singular joint; the output of this layer intends to position its corresponding joint in a previously specified angle, but it is filtered by the second layer of controllers $(f)$. Each motor is driven by a different function, which calculates the control action based on all the signals originated in the previous layer.

The controllers of the first layer are related with the dynamics of the link, they must be established once. The second layer of functions $(f)$ must be designed facing the DT intended to be completed. Functionally and computationally speaking the architecture is layered; every time step two mappings take place, one for error position reasons and the other seeking coordination between joints.

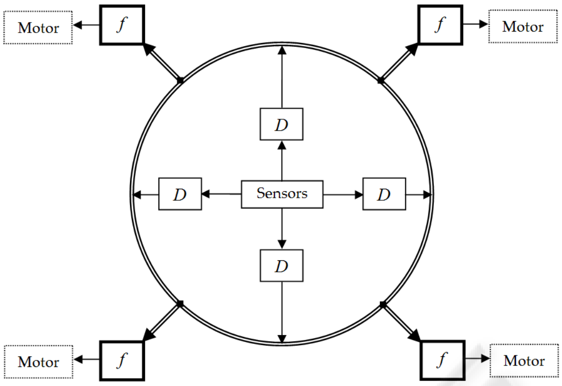


Figure 1: LCC Basic Idea : Two layers of control.

Every controller of the second layer $(f)$ uses the information originated in all the first layer's controllers. This collaboration allows joint controllers to know the dynamical state of the others, and then to act according to it in order to complete the DT.

### 2.1 Architecture Formulation

Let the articulated mobile robot have $n$ joints. Each joint has an error based controller $D_i$ with $(i = 1, ..., n)$ in the first layer. This is a SISO function, whose input is the error of the position of its respective joint $(e_i)$ and whose output is a control action $(u_i)$. The error is calculated using the sensor information providing the actual position of the joint $(\theta_i)$ and the already known set point or goal configuration $(\theta_i^*)$.

$$\begin{aligned} e_i &= \theta_i - \theta_i^* \\ u_i &= D_i(e_i) \qquad i = 1, ..., n \end{aligned}$$

Let the dynamical task have associated $n$ controllers $f_i$ in the second layer with $(i = 1, ..., n)$, these are MISO functions whose inputs are the $n$ outputs $u_i$ of the first layer described above. The output of the second layer functions $v_i$ is the velocity applied on the corresponding motor $(M_i)$.

$$v_i = f_i(u_1, ..., u_n) \qquad i = 1, ..., n$$

The robot state is represented by a continuous time dynamical system controlled by a parameterized policy,

$$\begin{aligned} \dot{x} &= g(x,v) + h(x) \cdot v && (1) \\ v &= \pi_w(x,v) && (2) \end{aligned}$$

where vector $x$ represents the states of the robot, $g$ and $h$ its dynamics, $v$ the control action and $\pi$ the control policy parameterized by the vector $w$.

### 2.2 LCC Synthesis

Once the control scheme has been stated, the next step is the definition of a methodology to compute the functions $(f_i, D_i)$. Here a policy gradient reinforcement learning (PGRL) algorithm is employed. PGRL methods (see (Williams, 1992),(Sutton et al., 2000)) are based on the measurement of the performance of a parameterized policy $\pi_w$ applied as control function during a delimited amount of time. In order to measure the performance of the controller, the following function is defined,

$$V(w) := J(x, \pi_w(x)) \tag{3}$$

where the measurement of the performance $V$ of the parameters $w$ is done by defining the *cost function* $J$.

By restricting the scope of the policy to certain class of parameterized functions $u = \pi_w(x)$, the performance measure (3) is a surface where the maximum value corresponds to the optimal set of parameters $w \in \mathbf{R}^d$. The search for the maximum can be performed by standard gradient ascent techniques,

$$w_{k+1} = w_k + \eta \nabla_w V(w) \tag{4}$$

where $\eta$ is the step size and $\nabla_w V(w)$ is the gradient of $V(w)$ with respect to $w$. The analytical formulation of this gradient is not possible without the acquisition of a mathematical model of the robot. The numerical computation is also not evident, then, a stochastic approximation algorithm is employed : the 'weight perturbation' (Jabri and Flower, 1992), which estimates the unknown gradient using a Gaussian random vector to orientate the change in the vector parameters. This algorithm is selected due to its good performance, easy derivation and fast implementation; note that the focus of this research is not the choice of a specific algorithm, nor the development of one, but rather the cognitive architecture to provide robots with the coordination learning ability.

This algorithm uses the fact that, by adding to $w$ a small Gaussian random term $z$ with $E\{z_i\} = 0$ and $E\{z_i z_j\} = \sigma^2 \delta_{ij}$, the following expression is a sample of the desired gradient

$$\alpha(w) = [J(w+z) - J(w)] \cdot z \tag{5}$$

Then, both layers' controllers can be found using this PGRL algorithm.

$$\begin{aligned} u_i &= D_{k_i}(e_i) \qquad (6) \\ v_i &= f_{w_i}(u_1, ..., u_n) \end{aligned}$$

In order to be consequent with the proposed definition of each layer, the training of the vector parameters must be processed independently; first the dynamical layer and then the coordination one. It is assumed that the movement has to take place between a limited amount of time $T$, where signals are collected to compute the cost function (3), then, the gradient is estimated using (5) and a new set of parameters obtained applying the update strategy of (4). Notice that in the case of the first layer, each function $D_{k_i}$ is trained and updated separately from the others of the same layer, but when learning the coordinate layer, all the functions $f_{w_i}$ must be updated at the same time, because in this case the performance being measured is that of the whole layer.

## 3 LINEAR IMPLEMENTATION

The previous description of the LCC is too general to be of practical use. Then, it is necessary to make some assumptions about the type of functions to be used as controllers. A well known structure to control the dynamic position of one link is the Proportional Integral Derivative (PID) error compensator. It has the following form,

$$u_i = K_{P_i} \cdot e_i + K_{D_i} \cdot \frac{de_i}{dt} + K_{I_i} \cdot \int e_i \, dt \tag{7}$$

The functionality of its three terms $(K_P, K_D, K_I)$ offers management for both transient and steady-state responses, therefore, it is a generic and efficient solution to real world control problems. By the use of this structure, a link between optimal control and PID compensation is revealed to robotic applications. Other examples of optimization based techniques for tuning a PID are (Daley and Liu, 1999; Koszalka et al., 2006).

The purpose of the second layer is to compute the actual velocity to be applied on each motor by gathering information about the state of the robot while processing a DT. The PID output signals are collected and filtered by this function to coordinate them. Perhaps the simplest structure to manage this coordination is a gain row vector $W_i$, letting the functions in the second layer be the following,

$$f_{w_i}(\mathbf{u}) = \mathbf{W}_i \cdot \mathbf{u}$$

Then, a linear combination of $\mathbf{u}$ commands the coordination. The matrix $W_{DT_m}$ encapsules all the information about this layer for the $m$th DT.

$$\mathbf{W}_{DT_m} = \begin{bmatrix} w_{11} & \dots & w_{in} \\ \vdots & \ddots & \vdots \\ w_{n1} & \dots & w_{nn} \end{bmatrix} \tag{8}$$

Where the term $w_{ij}$ is the specific weight of the $j$th PID in the velocity computation of the joint $i$.

# 4 HUMANOID COORDINATION

Classical humanoid robot motions are typically very slow in order to maintain stability; e.g., the biped walking based on Zero Moment Point (ZMP) (Vukobratovic and Stepanenko, 1972) condition avoids those states far from the pre-determined trajectory in which the momentum is guaranteed. By doing this, thousands of trajectories and capabilities of the robot could be restricted. Highly dynamic capabilities need those avoided states to perform quick and realistic motions.

## 4.1 The Simulation Robot Environment

A simulated humanoid is employed as a test-bed to evolve the controllers. It was modeled using Webots (Michel, 2004), a 3D robot simulation platform . Details can be obtained in (Webots, ).

This particular model of the full body humanoid robot has a total of 25 DOF, each link torque is limited to $[-10, 10]$ Nm, additionally it has a camera, a distance sensor, a gps, 2 touch sensors in the feet, and a led in the forehead. The model includes physical characteristics; the software processes kinematic and dynamic simulation of the interaction between the robot and its environment.

## 4.2 The One-Leg Equilibrium Task

The goal of the simulation experiment is to reach a final configuration called *'one-leg equilibrium point'* starting from the passive stand up position. Three joints are involved in this motion (see Figure 2(a).): The longitudinal axis of the Back, the transversal axis of the left hip and the left knee; their goal states, i.e. angles in radians, are $\theta_1^* = 0.4, \theta_2^* = 1.5$ and $\theta_3^* = 1.5$ respectively, with final velocity $\dot{\theta}^* = 0$ for all the joints. Both configurations, initial and final, are shown in Figure 2(b). The goal configuration has been designed to generate a slight inclination of the back to the right side, changing the center of mass of the body of the robot, thus, letting more weight supported on the right leg, and then allowing the left leg to lift, by means of the blending of the hip and knee articulations.

Figure 3 shows the simulation sequence of the motion of the robot using standard low level controllers. The robot falls down when attempts to reach the goal configuration due to the forces generated by the friction between the floor and its foot soles. The whole body of the robot suffers a destabilization at the start of the movement.

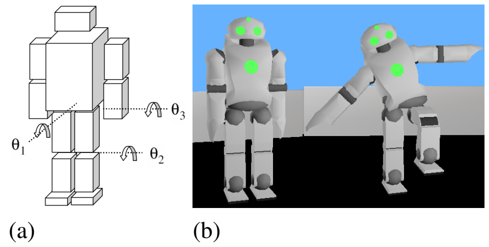


Figure 2: Simulated Humanoid (a). Joints involved in the motion. (b) Initial and goal configuration.

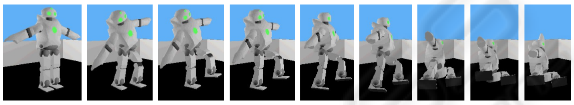


Figure 3: Direct Solution: Failure Demonstration.

## 4.3 The Learning Procedure

Table 1 shows the PGRL algorithm. A supervisor program is in charge of the training; its EVALUATE subroutine delivers the set of values to be evaluated in the robot controllers. It starts a controlled episode from the initial configuration and intends to achieve the goal states; every evaluation takes $T = 12s$, within this time the cost function is measured and returned. The supervisor repeats this operation until a convergence criterium is matched or the number of iterations is too large.

In this experiment the PID values are learned but not calculated. The following is the discrete implementa-

Table 1: Weight perturbation PGRL Algorithm.

```
input
  step size $\eta \in [0, 1]$
  search size $\sigma \geq 0$
  max step $\Delta_{max} \in [0, 1]$
initialize
  $W_0 \leftarrow \mathbb{I} \in \mathbf{R}^{3\times3}$ (Identity Matrix)
  $\alpha \leftarrow 0$
  $J \leftarrow$ EVALUATE($\mathbf{W}$)
repeat
  1. $z \leftarrow N(0, \sigma)$
  3. $J_z \leftarrow$ EVALUATE($\mathbf{W+z}$)
  4. $\alpha \leftarrow [J_z - J] \cdot z$
  5. $\Delta W \leftarrow -\eta\alpha$
  6. $\Delta W \leftarrow \min(\Delta W, \Delta_{max})$
  7. $W \leftarrow W + \Delta W$
  8. $J \leftarrow$EVALUATE($\mathbf{W}$)
until convergence or number of iterations too large
return $W, J$
```

tion of the PID employed,

$$u_i^t = K_P \cdot e_i^t + K_D \cdot (e_i^t - e_i^{t-1}) + K_I \cdot \sum_{j=0}^{t} e_i^j \qquad (9)$$

Where $u_i^t$ represents the control signal associated with joint $i$ at step time $t$, it depends on its corresponding position error $e_i^t = \theta_i^* - \theta_i^t$. This controller must provide zero position error in a single joint motion, therefore its parameters are learned using as performance criteria the following expression,

$$J_i(V) = -\sum_{t=0}^{T}(\theta_i^* - \theta_i^t)^2 + \sum_{t=0}^{T}(\dot{\theta}_i^t)^2 \qquad (10)$$

The PID is tested using a time step of $\Delta T = 32ms$ during $T = 12s$. The algorithms constants used are $\sigma = 0.032$ and $\eta = 0.05$. For the learning of the particular PID in charge of controlling the back, special restrictions are implemented to the reward function, penalizing those episodes where the robot falls down. The penalization is proportional to the time the robot stays on the ground.

The linear parameterized function selected for the coordination layer is:

$$\mathbf{v_t} = \mathbf{W} \cdot \mathbf{u_t} \qquad (11)$$

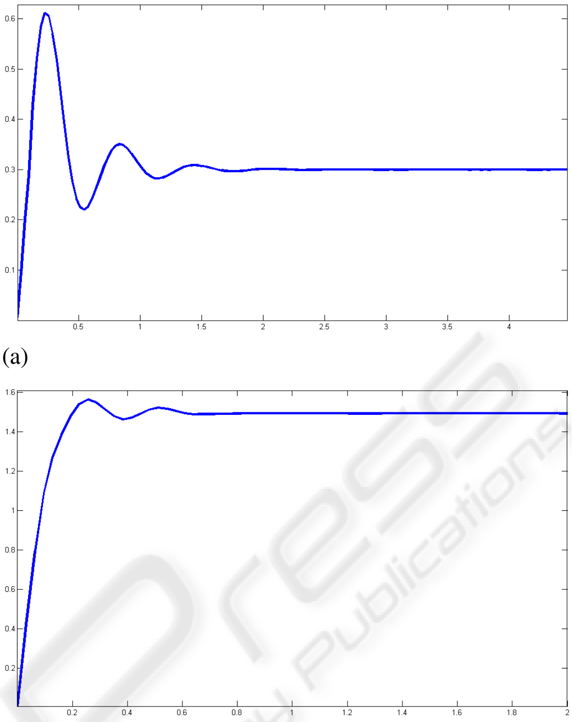
where $v_t \in \Re^3$ are the actual final velocities; $u_t \in \Re^3$ the outputs of the PIDs; and $W \in \Re^{3\times 3}$ is the transformation that contains the parameters to be found ($W_{ij}$).

The algorithm starts with a diagonal matrix as coordination controller, i.e. $W_0 = I$, meaning that the first collaborative control function attempted is that in which just the original PID controllers act. The non-diagonal values of $K$ deliver coordination. Same algorithms constants and time step are used in this stage of the LCC implementation.

# 5 RESULTS

## 5.1 PID Controller

Several local minima are found. Figure 4(a) shows the output of the back joint using one of the PID solutions found. It presents big overshoot, but no stationary error to a set point of $\theta^* = 0.3$, and mainly: The robot learns how 'not to fall'.
The behavior of the Back joint is equivalent with an inverted pendulum system with a big mass being manipulated (robot torso). For this joint, and for a better controlled output, a more complex parameterized policy would be needed. On the other hand, Figure 4(b) presents the output of the hip joint. It is evident that a better solution is achieved, this is because the training of this PID is done in a lay down position, with no risk of falling, and the mass of the link being controlled is widely insignificant compared with that in the back-joint case.



(a)

(b)

Figure 4: **(a)** Back-Joint position output (radians), using a learned PID (set point=0.3rad), **(b)** Hip-Joint position output (radians) using a learned PID (set point=1.5rad).

Final values for the independent joints controllers (Back, Hip and Knee) are :

$$\begin{aligned} PID_1 &= [0.746946 \;\; 0.109505 \;\; 0.0761541] \\ PID_2 &= [0.76541 \;\; 0.0300045 \;\; 0] \\ PID_3 &= [0.182792 \;\; 0.0114312 \;\; 0.00324704] \end{aligned}$$

## 5.2 Coordination Layer

After several iterations, the best solution is found; a sequence of the humanoid learned motion is depicted in Figure 5. The associate coordination controller is the following:

$$W = \begin{bmatrix} 0.6612900 & -0.3643360 & -0.1936260 \\ -0.0497653 & 1.0266600 & -0.0684569 \\ 0.0285586 & 0.1469820 & 0.9642390 \end{bmatrix}$$

Note how the non-diagonal parameters have grown, allowing the coordination. This characteristic has

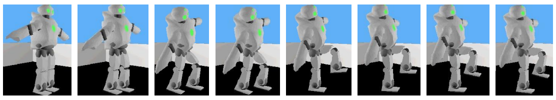


Figure 5: Simulated Humanoid One-Leg Equilibrium task.

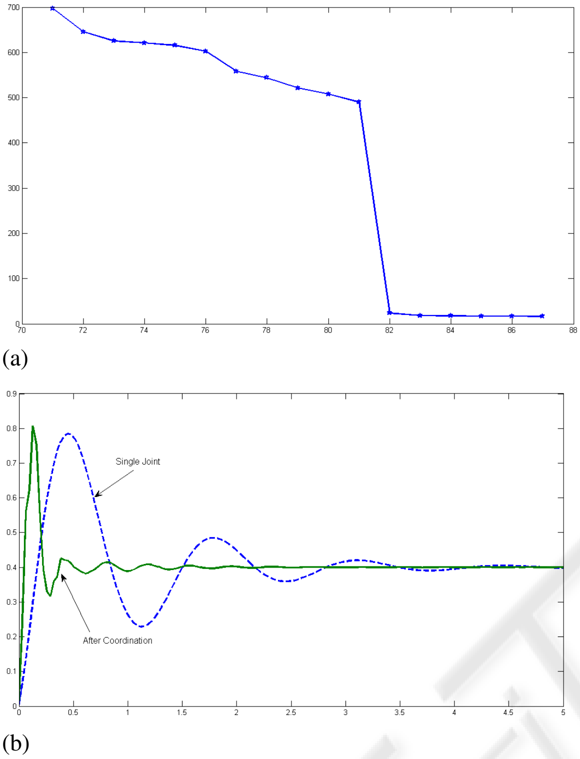


Figure 6: **(a)** Reward Evolution $J(W)$ of the best learned solution (last 20 tirals); **(b)** Back-joint position, before (dotted line) and after the coordination learning.

been previously pointed out in (Rosenstein and Barto, 2001), but here in an unstable 3D system task.

For this particular solution, the evolution of the reward is presented in Figure 6(a). Once the robot learns how 'not to fall' the convergence rate increases and the final results get to lower values.

It is important to mention that the robot presents some absolute displacement, tending to go backwards. A better designed reward function is needed to avoid absolute displacements, but a good solution is harder to find in this scenario. Finally, it is important to emphasize the change in the behavior of the back joint after the second layer phase. It seems like the system learns that the overshoot must be shorter in time in order to be able to lift the leg with out falling down. Figure 6(b). shows the output of the back joint before and after the second layer of learning.

# 6 CONCLUSIONS

The LLC scheme proposes to use the dynamic interaction of the robots articulations as a trajectory generator, without the use of a trajectory planner. Exploiting dynamics gives the robot the ability to explore motion interactions that result in optimized behaviors: *Learning at the level of dynamics to succeed in coordination*.

The presented solution overcomes the model dependency pointed above; by the presentation of a systematic control scheme the ideas of (Rosenstein and Barto, 2001) are extended. Here, the low level controllers are found using learning techniques and the formulation of a control architecture allows the implementation of different parameterized policies.

The interaction between Machine Learning, Control Systems and Robotics creates an alternative for the generation of artificial systems that consistently demonstrate some level of cognitive performance.

Currently, highly dynamic motions in humanoids are widely unexplored. The capability of this robots will be extended if motions that compromise the whole-body dynamic are explored. The LCC is tested within a simulated humanoid and succeed in the performance of a very fast motion, one in which the whole-body equilibrium is at risk.

Coordination is possible thanks to the sharing of information.

# ACKNOWLEDGEMENTS

This work has been partly supported by the Spanish MEC project ADA (DPI2006-15630-C02-01). Authors would like to thank Olivier Michel and Ricardo Tellez for their support and big help.

# REFERENCES

Daley, S. and Liu, G. (1999). Optimal pid tuning using direct search algorithms. *Computing and Control Engineering Journal*, 10(2):251–56.

Fujita, M. and Kitano, H. (1998). Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robots*, 5(1):7–18.

Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. (1998). The development of honda humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*.

Jabri, M. and Flower, B. (1992). Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks.

*IEEE Transactions on Neural Networks*, 3(1):154–157.

Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., and Isozumi, T. (2004). Humanoid robot hrp-2. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*.

Koszalka, L., Rudek, R., and Pozniak-Koszalka, I. (2006). An idea for using reinforcement learning in adaptive control systems. In *Proceedings of the IEEE Int Conference on Networking, Systems and Mobile Communications and Learning Technologies*, Kerkrade, Netherlands.

Kuroki, Y., Blank, B., Mikami, T., Mayeux, P., Miyamoto, A., Playter, R., Nagasaya, K., Raibert, M., Nagano, M., and Yamaguchi, J. (2003). A motion creating system for a small biped entretainment robot. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, IROS*.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42.

Rosenstein, M. T. (2003). *Learning to exploit dynamics for robot motor coordination*. PhD thesis, University of Massachusetts, Amherst.

Rosenstein, M. T. and Barto, A. G. (2001). Robot weightlifting by direct policy search. In *Proceedings of the IEEE International Conference on Artificial Intelligence, IJCAI*, pages 839–846.

Sutton, R., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12:1057–1063.

Vukobratovic, M. and Stepanenko, J. (1972). On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15:1–37.

Webots. http://www.cyberbotics.com. Commercial Mobile Robot Simulation Software.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.