A General Control Architecture for Dynamic Bipedal Walking

Chee-Meng Chew and Gill A. Pratt
MIT Leg Laboratory, Cambridge, MA 02139
http://www.ai.mit.edu/projects/leglab/

Abstract

We propose a general but simple bipedal walking control architecture that incorporates intuitive control and learning algorithms. The learning algorithm is mainly used to generate the key parameters for the swing leg. The intuitive control is used to maintain the height and body posture. Based on the proposed architecture, a control algorithm is constructed and applied to a planar biped and a 3D biped. By applying appropriate local speed control mechanism, we demonstrate that the bipeds can successfully achieve walking of 100 *seconds within a reasonable number of trials. No dynamic models or nominal joint trajectory data are required for the implementations.*

1 Introduction

Pratt et al. [15, 13, 5] had demonstrated that simple intuitive control algorithms can be applied to bipedal walking which can be difficult to analyze mathematically. To simplify the bipedal walking task, they partitioned it into three subtasks: 1) height control; 2) body posture control; and 3) walking velocity control. It is very easy to achieve the first and second subtasks. The third subtask is more complex and usually requires several control mechanisms.

There are a number of velocity control mechanisms. They can be classified into three main types: 1) global control; 2) local control; and 3) strategy based. The *global* control consists of the swing leg control. It is called *global* because it can achieve velocity control of the biped by itself. *Local* control approaches have only secondary effect on the walking velocity. Though they may help to improve the profile of the walking velocity, they may be insufficient to rectify the situation where the swing leg is badly controlled. Examples of the local velocity control mechanisms are those based on the double support phase [13, 5], the biped's torso, the stance ankle etc..

The strategy based approaches are abundant and one example is the addition and removal of energy at strategic time [15]. Strategy based approach can be powerful but usually requires intuition. However, we do not use any velocity control strategies in this paper.

It is clear from the taxonomy of the velocity control mechanisms that swing leg based control is the key to walking velocity control because it affects the global walking velocity. The swing leg behavior is determined by any combinations of these approaches: 1) prescribing the swing leg trajectory in the joint or Cartesian space; 2) natural dynamics [16]; 3) strategy-based. Here, we mainly adopt the first approach.

In biomechanics area, most researches on swing leg analysis typically adopted a trial-and-error model fitting approach [18, 20, 2]. However, application of any successful model, for example in human walking analysis, may not be directly applicable for the control of a bipedal robot. It is because the robot usually does not have the same degrees of freedom as human beings, nor does it have the same actuators system and mass distribution.

Although the swing leg motion is an important determinant for the stabilization of the walking speed and the transverse motion, it is extremely hard to obtain a close form expression for it. It is because the dynamic equations of a biped are complex and nonlinear. This is further complicated by limited or no actuation at the stance's ankle and the discontinuities caused by the impact at the support exchange. The overall motion of the system in the sagittal plane is usually strongly coupled to the swing leg motion in the same plane unless the leg's inertia is negligible or the motion is slow. To solve the problem, we would like to apply a learning approach for the swing leg behavior.

In this paper, we synthesize a general control architecture for bipedal walking based on an intuitive control algorithm that is assisted by learning. Learning is mainly used for the swing leg behavior.

We implement a control algorithm based on the architecture. An intuitive control approach called *Virtual Model Control* (VMC) [13, 5] is used to design an algorithm for subtasks like height and body posture control. Reinforcement Learning (RL) methods [7, 19] will be used as the learning tool. A biped learns the key swing leg parameters so as to achieve stable walking gait.

The control algorithm is expected to have the following properties:

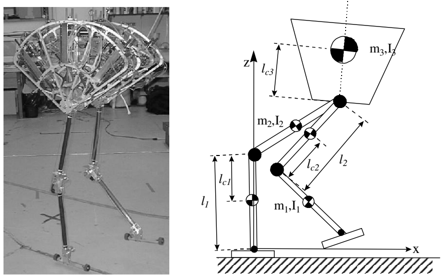
1. easy to implement (minimum manual tuning);
2. suitable for real-time application;
3. applicable across bipeds with different actuator behavior, mass and length parameters;
4. does not require nominal trajectories or model;
5. able to achieve variable walking speed;
6. able to control step length.

2 Bipedal Robots

We consider two simulated bipedal robots in our analysis. One of them is a computer model of a headless and armless planar bipedal robot called *Spring Flamingo* (see Figure 1). The legs are much lighter than the body. The total mass is about 12 kg (legs' mass is about 1 kg each). Each leg has three actuated rotary joints. The joints axes are perpendicular to the sagittal plane. The maximum hip height is 0.84 m.

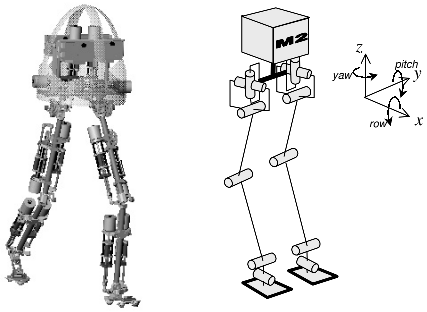
The other is a computer model of a 3D biped called *M2* (Figure 2). It is also headless and armless. It has six actuated degrees of freedom on each leg as shown in Figure 2. However, the proportion of legs' mass (about 6.5 kg each) over the total mass (about 25 kg) is higher than that of the former. The maximum hip height is 0.915 m.

In both physical systems, *Series Elastic Actuators* [12] are used to generate torque outputs at the joints based on the desired torque from the control algorithm. In the simulation, we assume that the joints are equiped with torque sources.



(a) (b)

Figure 1: 7-Link 6 dof planar biped - Spring Flamingo



(a) (b)

Figure 2: Three dimensional biped: M2 (CAD drawing by Daniel Perluska)

3 Control Architecture

The schematic diagram of the proposed control architecture is shown in Figure 3. The intuitive control algorithm can be implemented in many different ways. In this paper, we adopt one that is based on the Virtual Model Control approach (Subsection 3.1). The algorithm includes a state machine that consists of two main states: 1) single support phase; and 2) support exchange phase.

Some of the swing leg's parameters are learned by trial-and-error using Reinforcement Learning (see Subsection 3.2). The biped has no prior data or nominal model to generate a nominal behavior that results in stable walking. The learning agent will select the appropriate swing leg parameters, for example, step length, swing time etc.. Then, the swing leg parameters are sent to the intuitive control algorithm to generate the desired joints' torque. The learning agent evaluates the outcome and update itself accordingly.

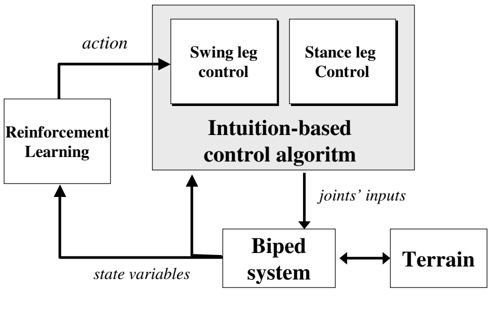


Figure 3: Proposed control architecture

One characteristic of the resulting control algorithm is its low computation requirement which is inherited from the Virtual Model Control approach and the selected Reinforcement Learning algorithm. Thus, it can be implemented in real-time.

3.1 Virtual Model Control

Virtual Model Control [14, 13] uses virtual mechanical components to generate actuators' torque (or force) commands. The joints' torque creates the same effect that the virtual components would have created, had they existed, thereby creating the illusion that the simulated components are connected to the real robot. Such components can include simple springs, dampers, masses, etc.

3.2 Reinforcement Learning

A characteristic of Reinforcement Learning (RL) paradigm is that learning is done by interacting with the environment [19]. This is distinct from a supervised learning approach where selected input-output data set is used to train a learner.

One nice feature of the Reinforcement Learning approach is that all non-linearities like bandwidth limitation of the actuators, torque limitation, etc. are encapsulated from the viewpoint of the learner.

There are many algorithms proposed for Reinforcement Learning (RL). For control application, RL algorithms usually learn an intermediate function (called value function) from which decision or control action is deduced. We adopt one such algorithm called *Q-learning* [21]. It is an off-policy and a model-free RL algorithm. A brief outline of the algorithm is given as follows.

Let i denote the state; u denote the action; and $Q(i, u)$ be the estimated state-action value (or Q-value) corresponding to i and u. The key equation of the Q-learning algorithm is an update equation as follow [19]:

$$\begin{aligned} Q(i_t, u_t) \leftarrow & Q(i_t, u_t) + \\ & \alpha[r_{t+1} + \gamma \max_u Q(i_{t+1}, u_{t+1}) - Q(i_t, u_t)] \end{aligned} \tag{1}$$

where the subscript indicates the stage number; r_{t+1} denotes reward received for a transition from state i_t to i_{t+1} after an action u_t is taken at t-th stage; α denotes a step-size parameter for the update, γ denotes a discount rate in the computation of cumulative rewards (return). Equation 1 updates the estimated state-action value $Q(i_t, u_t)$ based on the immediate reward r_{t+1} and the maximum estimated state-action value $Q(i_{t+1}, u)$ over all feasible u.

To apply Q-learning to continuous state and action, a function approximator is used to approximate $Q(i, u)$. A function approximator usually has generalization property that helps to accelerate the learning process. Possible function approximators are the Gaussian Radial Basis Function (GRBF) network [4, 11] and the Cerebellar Model Articulation Controller (CMAC) [1, 9] because they have localized generalization property. Although the GRBF Network has localized generalization and is a universal approximator [10], its implementation is computationally expensive. On the other hand, CMAC network has the advantage of having not only localized generalization, but also low in computation.

4 Implementations and Simulation Results

We assume that the walking motion in the sagittal, frontal and transverse planes are independent. Thus, we consider each of them separately. This section describes the detailed algorithms for the sagittal and the frontal plane to achieve dynamic walking of the simulated bipeds on level ground. We set the desired walking height, body posture and walking speed to be constant. The walking pattern is like those of the birds.

4.1 Sagittal plane algorithm

The height control and body posture control subtasks can be achieved using Virtual Model Control without the need of learning. For the stance leg control in the sagittal plane, we use an algorithm by Pratt et al. [13] for single support phase with limp stance ankle. In the algorithm, the height control is achieved by a virtual parallel spring-damper attached vertically between the hip and the ground. This generates a virtual vertical force F_z at the hip. The body posture (pitch) control is achieved by a virtual rotational spring-damper applied between two frames with common origin. One of the frames is attached to the body and the other has fixed orientation with respect to the inertia frame. This generates the virtual torque M_{pitch} about the hip. These virtual forces can then be transformed into the joint space using a transformation matrix [14].

For the forward velocity control, instead of using a double support phase as in [14, 5], we use the Reinforcement Learning algorithm to learn the right parameters for the swing leg. The reason of using learning is that there is no general analytical solution or good heuristics of the relationship between the swing leg's parameters and desired walking speed.

For the RL algorithm, the following variables are identified to be the state variables:

1. Velocity of the hip in the x-direction, $\dot{x}^+$;

2. x-coordinate of the swing ankle measured with reference to the hip, x^+_{ha} ;

3. Actual step length, l_{step} .

Superscript + indicates that a state variable is measured or computed momentarily after the landing of the swing foot.

We fixed the end position of the swing foot but learn the swing time. The end position of the swing foot can be measured from different references. For example, we can specify the end position relative to the hip. Another choice is to simply specify the desired step length. The latter is desirable for walking over difficult terrain like stairs where the location of the foot placement needs to be more precise. However, the former contains simple intuition. For example, we want the swing foot to land much ahead of the hip if we want to slow down, and the opposite for speeding up.

We choose a simple reward function as follows:

$$r = \begin{cases} 0 & \text{for } -0.1 < \dot{x} < 1.3m/s, \\ -1 & \text{otherwise (failure).} \end{cases} \tag{2}$$

We also assign $r = -1$, if the robot falls down.

Q-learning using CMAC network as the function approximator is used in this implementation. α and γ of Equation 1 are assigned 0.1 and 0.9 respectively.

At each step, Q-learning generates a value for the swing time from a predefined discrete set (0.2 to 1.0 second, increment of 0.02 second). For the swing foot strategy, it consists of three distinct behaviors: (1) lift-up; (2) swing-forward; and (3) landing. The trajectories for the lift-up and swing-forward behaviors are each generated by a third degree polynomial function of time. The landing segment is achieved naturally by the gravitational force. The total time from the beginning of the lift-up to the end of the swing-forward behavior corresponds to the swing time.

No exploration is included in the algorithm because it is undesirable for such a control problem. The biped tries to walk until it has failed. Each time it fails, it will restart a new trial. Learning is stopped when the biped has completed a target time (100 seconds) of walking.

4.1.1 Usage of local control to improve learning rate

In the simulation analysis, we observe that the learning rate of the system can be improved tremendously when proper local control is applied to the stance leg's ankle to assist in regulating the walking speed. We identify one such control law as follows:

$$\tau_a = B_a(\dot{x} - \dot{x}^d). \tag{3}$$

where τ_a is the ankle torque, B_a is a constant coefficient and superscript d indicates a desired value. The torque τ_a is bounded within upper and lower limits to prevent the stance leg's foot from tipping at the toe or heel.

To study the effectiveness of the local control, we run several simulations on the *Spring Flamingo* model, each of them having different initial conditions and the learning agent tries to learn the swing time given the desird foot placement with respect to the hip. They show consistent results that the stance ankle control indeed helps to improve the learning rate tremendously even though the feet are only $15cm$ long. We show two learning curve in Figure 4, one with the local control (dotted graph) and the other without (solid-line graph). The initial posture (legs are parallel to each other) and initial walking speed ($0.4m/s$) are the same for every trial. The desired walking speed is $0.7m/s$.

The dotted graph shows that the simulated biped reached 100 seconds walking in less than 30 iterations. The solid-line graph corresponds to the case that had limp stance ankle. The simulated biped did not achieve 100 seconds walking even after more than 400 iterations.

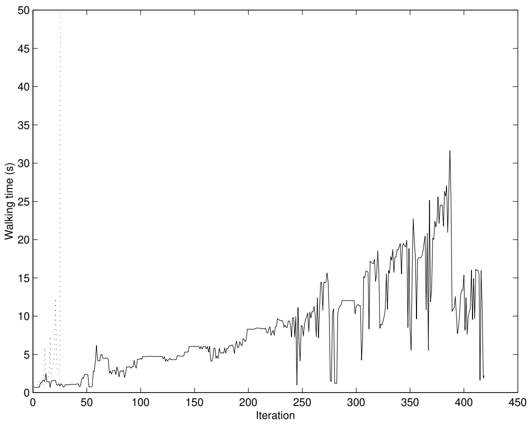


Figure 4: Local control at the stance leg's ankle joint increases the learning rate. The starting posture and initial speed are the same for both cases. Dotted curve corresponds to the simulation result with the local control at the stance ankle. Solid curve is without the local control.

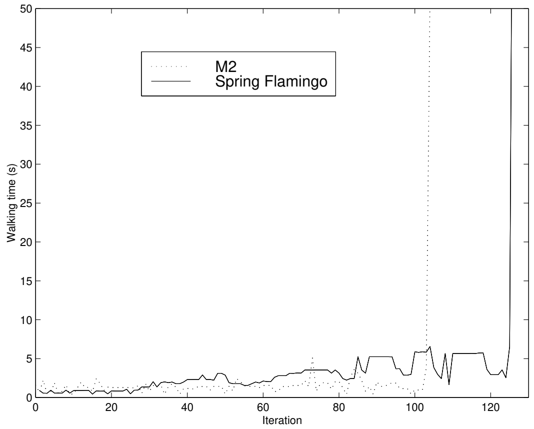


Figure 5: Learning curves for simulated M2 and Spring Flamingo.

4.1.2 Generality of the control algorithm

This subsubsection demonstrates the generality of the control algorithm. The learning agent is used to learn the swing time given the desired step length which is kept constant ($0.3m$). We apply the algorithm to both simulated *Spring Flamingo* and *M2* (planar). The initial posture and hip velocity are the same for each trial. The desired hip velocity is $0.7m/s$ for both bipeds. The desired heights are $0.74m$ for *Spring Flamingo* and $0.84m$ for *M2*. Again, we use the local control of the stance leg's ankle to accelerate the learning. The results are shown in Figure 5. The learning speeds are comparable and this demonstrates the generality of the control algorithm in term of its application to bipeds of different inertia and length parameters.

The simulation result for the Spring Flamingo implementation after the learning is shown in Figure 6. Those variables with "i" preceding them are the state variables for the Reinforcement Learning algorithm and that with "u" preceding it is the action. The top graph shows the state number. State 5 and 6 correspond to the right support and the left support respectively. State 7 corresponds to the transition from the right-support to the left-support and state 8 corresponds to the transition from the left-support to the right-support. We observe that the actual hip height z and hip velocity $\dot{x}$ are well behaved and they are close to the desired value (indicated by dashed lines).

4.2 Frontal plane algorithm and 3D walking implementation

In this plane, the body posture control subtask can be achieved using virtual rotational spring-damper as in the sagittal plane. Assuming that the swing time is determined by the sagittal plane implementation, the biped is left with the decision of lateral foot placement to maintain lateral balance. In M2, the lateral foot placement can be achieved by servoing the hip roll joint of the swing leg.

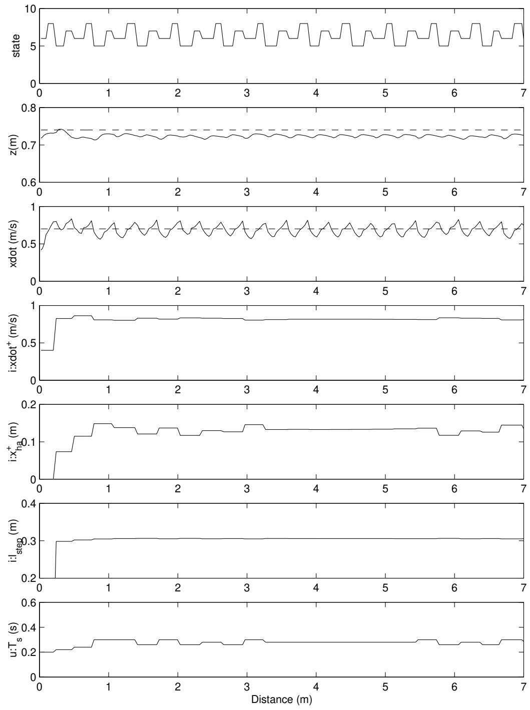


Figure 6: The simulation data for the dynamic walking of the simulated Spring Flamingo after the learning.

We first try to apply the RL algorithm to learn the target end position of the hip roll joint of the swing leg because there is no analytical solution for it. However, the result is not satisfactory though 100 seconds of walking is possible. The resulting motion in the frontal plane has chaotic behavior.

Nevertheless, the simulation analysis leads us to adopt an intuitive and powerful notion: *symmetry*. Raibert [17] has used it to design an algorithm for foot placement to control the forward speed of a hopping biped.

We derive an algorithm for the frontal plane motion based on the notion of symmetry. Assuming that both legs have similar actuators' and structural characteristics, the hip roll joint of the swing leg is commanded to mirror the behavior of the stance leg's joint.

To ensure that such an approach does not diverge, a local control at the stance ankle's roll joint is used.

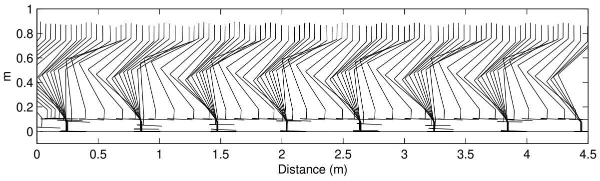


Figure 7: Stick diagram of the dynamic walking of the simulated M2 after the learning. Only the left leg is shown in the diagram and the images are 0.1 second apart.

The local control is chosen as follows:

$$\tau_{a\_roll} = -B_{a\_roll}\dot{y} \tag{4}$$

where τ_{a_roll} is the desired torque of the stance ankle's roll joint, B_{a_roll} is a constant coefficient and $\dot{y}$ is the lateral velocity of the hip.

To prevent the stance leg from getting into the way of the swing leg, we adjust the swing leg in the frontal plane depending on the current stance leg position.

We implement simulated 3D walking for M2 using the sagittal plane and the frontal plane algorithms developed in this paper. The desired forward velocity is set at $0.4m/s$. Learning is only applied to the sagittal plane algorithm. The simulated biped is able to achieve 100 seconds of walking within 100 iterations. The simulation result of the walking after the learning is shown in Figure 8. We observe that the body posture, the height (z), the forward velocity and the lateral velocity are well behaved. A stick diagram of the walking simulation after the learning is as shown in Figure 7.

5 Discussion

Although the local velocity control mechanisms have only secondary effect on the walking velocity, it may enhance the robustness of the walking gait. It also improves the quality of the velocity control that is not easily achieved by the global mechanism. The local control mechanism can also be viewed as some sort of "entrainment" mechanism during the learning of the global mechanism. It can significantly increase the rate to reach a goal for a learning process.

We illustrate how intuitive strategy can be effective in simplifying the bipedal walking implementation if one can find one. For example, the notion of symmetry removes the learning requirement for the frontal plane control even though no physical model is used.

The successful 3D walking implementation in the simulation reveals that the frontal plane and sagittal plane motion can indeed be treated independently.

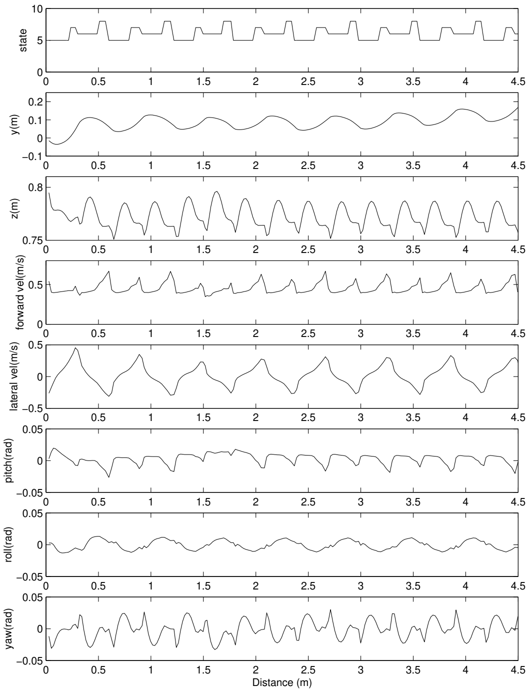


Figure 8: The simulation result (after learning) of the 3D walking of M2.

This implies that we can apply the past planar walking researches [13, 5] to 3D walking.

In the proposed architecture, the learning is localized to the swing leg behavior. It is the different from approach in [3] which adopted a "Melting pot" and modular approach when applying the Reinforcement Learning for the bipedal walking. In their implementation, the outcome of the learning system was the adjustment to the nominal joints' trajectories. Miller [6] also implemented a neural network learning approach that provided the desired joint trajectories for proper balance for side-to-side and front-to-back motion of a biped. One disadvantage of both approaches was that nominal trajectories of the joints were required before any learning began. It is also difficult to formulate tuning rules in the joint space due to the dynamic coupling especially during fast walking.

Some researches of bipedal walking assumed a massless leg model [8] and used local control at the stance's ankle to compensate for the disturbances due to the swing leg's dynamics and error in the model. The swing leg trajectory needed to be planned carefully so as not to create significant disturbances to the overall walking task. This approach was more suitable for low speed walking. Furthermore, it was not able to use the dynamics of the swing leg in the walking. In our approach, we do not have these restrictions.

To further reduce the learning time, we may use more heuristics to reduce the search space of the action and to detect failure earlier. We can also achieve it by using more advanced RL algorithms, for example, one that incorporates eligibility traces.

One indirect application of the proposed algorithm is that it can generate a successful walking pattern from which nominal trajectories or data can be extracted for other learning approaches. We may also apply reverse engineering to extract empirical rules, equations, or parameters to generate appropriate actions. It may also enable us to compare the effectiveness of different strategies of walking control by looking at the resulting performance (e.g. learning rate) in the algorithm.

6 Conclusions

We have constructed a simple control algorithm for dynamic bipedal walking based on the proposed control architecture. The algorithm has low computation and thus applicable for real-time implementation. We applied the control algorithm for both planar and 3D walking of the simulated bipeds. No model or nominal data were needed in the implementations.

We have demonstrated the generality of the algorithm by applying it to two simulated planar bipeds that have different mass and length parameters.

We have also demonstrated the application of local control approach in the sagittal plane to significantly reduce the number of iterations to achieve the walking goal.

By using the simple but powerful notion of symmetry, we have removed the learning need for the lateral foot placement. This results in a simple algorithm for the frontal plane motion control.

We will extend the algorithm so that the biped can learn to walk with variable speed. We will also modify the algorithm so that M2 can achieve human-like walking.

References

[1] J. S. Albus. *Brain, Behavior and Robotics*, chapter 6, pages 139–179. BYTE Books. McGraw-Hill, Peterborough, NH, 1981.

[2] R. Beckett and K. Chang. An evaluation of the kinematics of gait by minimum energy. *J. Biomechanics*, 1:147–159, 1968.

[3] H. Benbrahim and J. A. Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22:283–302, 1997.

[4] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2:321–355, 1988.

[5] Chee-Meng Chew, Jerry E. Pratt, and Gill A. Pratt. Blind walking of a planar bipedal robot on sloped terrain. *IEEE International Conference on Robotics and Automation*, 1999.

[6] Thomas W. Miller III. Real time neural network control of a biped walking robot. *IEEE Control Systems Magazine*, pages Feb:41–48, 1994.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, (4):237–285, 1996.

[8] S. Kajita and K.Tani. Experimental study of biped dynamic walking in the linear inverted pendulum mode. *IEEE International Conference on Robotics and Automation*, 1995.

[9] W. T. Miller, F. H. Glanz, and M. J. Carter. Cmac: An associative neural network alternative to backpropagation. *IEEE Proc.*, (78):1561–1567, 1990.

[10] P. C. Parks and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, pages 246–257, 1991.

[11] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. IEEE*, (78):1481–1497, 1990.

[12] Gill A. Pratt and Matthew M. Williamson. Series elastic actuators. *IEEE International Conference on Intelligent Robots and Systems*, 1:399–406, 1995.

[13] J. Pratt, P. Dilworth, and G. Pratt. Virtual model control of a bipedal walking robot. *IEEE International Conference on Robotics and Automation*, pages 193–198, 1997.

[14] Jerry E. Pratt. Virtual model control of a biped walking robot. Master's thesis, Massachusetts Institute of Technology, August 1995.

[15] Jerry E. Pratt and Gill A. Pratt. Intuitive control of a planar bipedal walking robot. *IEEE International Conference on Robotics and Automation*, 1998.

[16] Jerry E. Pratt and Gill A. Pratt. Exploiting natural dynamics in the control of a 3d bipedal walking simulation. *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR99)*, 1999.

[17] Marc. H. Raibert. *Legged Robots That Balance.* MIT Press, Cambridge, MA., 1986.

[18] M. S. Redfern and T. Schumann. A model of foot placement during gait. *J. Biomechanics*, 27(11):1339–1346, 1994.

[19] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[20] M. Townsend. Biped gait stabilization via foot placement. *J. Biomechanics*, 18(1):21–38, 1985.

[21] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.