Dynamic Bipedal Walking Assisted by Learning

by

Chee-Meng Chew

B.Eng., Mechanical Engineering
National University of Singapore, 1991

S.M., Mechanical Engineering
Massachusetts Institute of Technology, 1998

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2000

Author . . . .
Department of Mechanical Engineering
August 4, 2000

Certified by . . . .
Gill A. Pratt
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by . . . .
Kamal Youcef-Toumi
Professor of Mechanical Engineering
Thesis Committee Chair

Accepted by . . . .
Ain A. Sonin
Chairman, Department Committee on Graduate Students

Dynamic Bipedal Walking Assisted by Learning

by

Chee-Meng Chew

Submitted to the Department of Mechanical Engineering
on August 4, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

This thesis proposes a general control architecture for 3D dynamic walking. It is based on a divide-and-conquer approach that is assisted by learning. No dynamic models are required for the implementation.

In the approach, the walking task is divided into three subtasks: 1) to maintain body height; 2) to maintain body posture; and 3) to maintain gait stability. The first two subtasks can be achieved by simple control strategies applied to the stance leg. The third subtask is mainly effected by swing leg behavior. By partitioning the overall motion into three orthogonal planes (sagittal, frontal and transverse), the third subtask can be further divided into forward velocity control (in the sagittal plane) and lateral balance (in the frontal plane). Since there is no explicit solution for these subtasks, reinforcement learning algorithm, in particular, Q-learning with CMAC as function approximator, is used to learn the key parameters of swing leg strategy which determine the gait stability.

Based on the proposed architecture, different control algorithms are constructed and verified using the simulation models of two physical bipedal robots. The simulation analysis demonstrates that by applying an appropriate local control laws at stance ankle joints, the learning rates for the learning algorithms can be tremendously improved.

For the lateral balance, a simple control algorithm (without learning) based on the notion of symmetry is discovered. The stability of the control algorithm is achieved by applying appropriate local control law at the stance ankle joint. A dynamic model is constructed to validate this approach.

The control algorithms are also demonstrated to be general in that they are applicable to different bipeds that have different inertia and length parameters. The designer is also free to choose the walking posture and swing leg strategy. Both "bird-like" and "human-like" walking postures have been successfully implemented for the simulated bipeds.

Thesis Supervisor: Gill A. Pratt
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I am indebted to all my lab mates (Jerry Pratt, David Robinson, Dan Paluska, Allen Parseghian, John Hu, Mike Wessler, Peter Dilworth, Robert Ringrose, Hugh Herr, Chris Morse, Ben Krupp, Gregory Huang, Andreas Hoffman) for their contributions to my knowledge and awareness in my research area. They were never too tired to answer all my doubts and help me when I ran into problems. I also thank Jerry for designing and building the planar biped "Spring Flamingo". I also thank Jerry, Dave and Dan for designing the three-dimensional biped "M2". These bipeds allowed me to set an application target for my research. I would also like to thank all the present and former lab members who contributed to the design of the simulation package, especially Creature Library, without which my research would have been very tedious. I wish to thank my advisor, Professor Gill Pratt, for letting me take on this research. I also wish to thank my other thesis committee members (Professor Youcef-Toumi, Professor Jean-Jacques Slotine and Professor Tomaso Poggio) for coaching me so that I stayed on the right path. I would like to thank Gregory, Andreas, Jerry, Megan and Professor Pratt for proof-reading my thesis. I also wish to thank Professor Leslie Pack Kaelbling and Professor Dimitri P. Bertsekas for answering my questions in the area of reinforcement learning.

I would like to thank all the previous researchers in the biped locomotion for contributing many highly valuable research results based on which I organized my research. I also wish to thank all the researchers in the reinforcement learning field for designing excellent algorithms that can be applied to solve difficult control problems like bipedal walking.

I wish to thank the National University of Singapore because my work at MIT would never have happened without its sponsorship. I also wish to thank my parents, wife (Kris), daughter (En-Chin), and son (Ethan), for being very understanding and for providing me with moral support just at the right moment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Animal locomotion research has been around for more than a century. As defined by Inman, Ralston and Todd [41]: "Locomotion, a characteristic of animals, is the process by which animal moves itself from one geographic position to another. Locomotion includes starting, stopping, changes in speed, alterations in direction, and modifications for changes in slope."

Bipedal locomotion is associated with animals that use exactly two limbs to achieve locomotion. There are numerous reasons for studying bipedal locomotion and building bipedal robots. The main motivation is that human beings are bipedal. It is desirable for us to replicate a machine with the same mobility as ourselves so that it can be placed in human environments, especially in areas that pose great hazard for human beings.

It is also interesting to understand the basic strategies of how bipedal creatures perform walking or running. It is interesting to find out why the walking task, which can be easily performed by human beings and other living bipeds, seems to be daunting for a robotic biped. This understanding may aid the development of better leg prostheses and help many people who lose their lower extremities walk again. It may also be applied to rehabilitation therapy for those who lose their walking ability.

It is a great challenge for scientists and engineers to build a bipedal robot that has agility and mobility similar to a human. The complexity of bipedal robot control is mainly due to the nonlinear dynamics, unknown environment interaction, and limited torque at the stance ankle (due to the fact that the feet are not fixed to the ground).

Many algorithms have been proposed for the bipedal walking task [71, 25, 36, 32, 7, 43, 38, 39, 40, 52, 50, 55, 28, 70, 31]. To reduce the complexity of the bipedal walking analysis, some of the researchers restricted themselves to planar motion studies, some adopted simple models or linearization approaches, etc. Waseda University [106, 107, 122, 123] has conducted much research on 3D bipedal walking. All the implementations adopt a stabilization approach using trunk motion. The algorithms derived are thus not applicable to a biped that does not have the extra DOF on the body.

Honda humanoid robots (P2 and P3) [38] are the state-of-the-art 3D bipedal walking systems. They do not have extra DOF on the body to provide the stabilization control as in Waseda's robots. The lower extremities have the minimum number of DOF (6 DOF in each leg) which still allow the same mobility as human beings. Thus, the robots are capable of achieving most walking behaviors of their human counterparts.

Honda's control method is based on playing back trajectory recordings of human walking on different terrains [78]. Though the resulting walking execution is very impressive, such a reverse engineering approach requires iterative parameter tunings and tedious data adaptation. These are due to the fundamental differences between the robots and their human counterparts, for example, the actuator behaviors, inertias, dimensions, etc.

The MIT Leg Laboratory has recently designed and built a 3D bipedal robot called M2 (see Section 1.2). The robot has the same DOF at the lower extremities as the Honda's robots. Apart from the approach by Honda, there are very few algorithms proposed for such a biped. In this thesis, the author aims to construct and investigate new 3D dynamic walking algorithms for the biped.

1.2 Target Systems

For legged robots, the control algorithms are very much dependent on the structure, degrees of freedom, actuator's characteristics of the robot. This section describes the bipedal robots for which the walking control algorithms are designed.

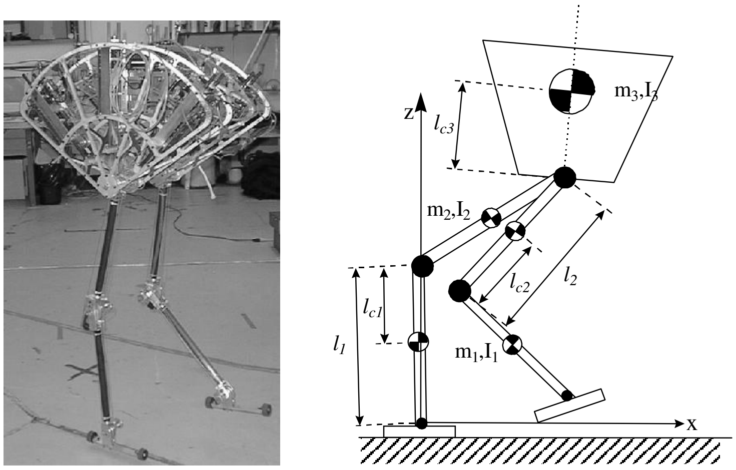
Two bipedal robots will be considered in this thesis. One is a headless and armless 7-link planar bipedal robot called Spring Flamingo (see Figure 1-1). It is constrained to move in the sagittal plane. The legs are much lighter than the body. Each leg has three actuated rotary joints. The joint axes are perpendicular to the sagittal plane. The total weight is about 12 kg.

The biped has a rotary angular position sensor at each DOF. It is used to measure the relative (anatomical) angular position between consecutive links. In the simulation model, it is assumed that the biped has a single axis gyrocope or an inclinometer fixed to the body which can provide the body posture information [1]. Each foot has two contact sensors at the bottom, one at the heel and one at the toe, to determine ground contact events.

The other robot is an unconstrained (3D) biped called M2 (Figure 1-2). It is also headless and armless. However, it has more massive legs compared with Spring Flamingo. Each leg has six active DOF of which three DOF is available at the hip (yaw, roll, pitch), one at the knee (pitch) and two at the ankle joint (pitch, roll). Each DOF has an angular position sensor to measure the relative angle between two adjacent links. Each of the feet has four force sensors (two at the toe and two at the heel) to provide the contact forces between the feet and the ground. A gyroscopic attitude sensor is mounted in the body to detect its roll, pitch and yaw angles. The total weight is about $25kg$.

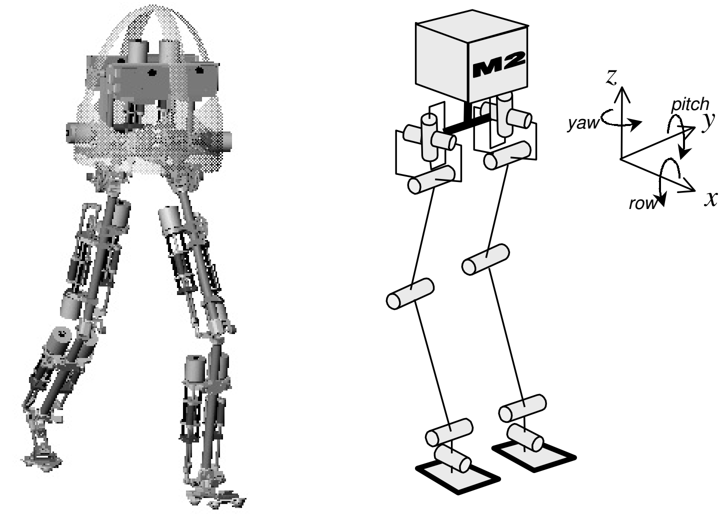
In both systems, the joints are driven by force/torque controlled actuators called Series Elastic Actuators [79]. The dynamic range of the actuators is quite good, and they are assumed to be force/torque sources in the simulation analysis in this thesis. Simulation models of these robots are created using the simulation tool described in Section 1.5. The model specifications for Spring Flamingo and M2 are given in Table 1.1 and 1.2, respectively.

[1] In the real robot, the body posture is provided by an angular position sensor attached between the body and a boom.



(a) (b)

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



(a) (b)

Figure 1-2: Three-dimensional biped: M2 (CAD drawing by Daniel Paluska)

Table 1.1: Specifications of Spring Flamingo

Description	Value
Total Mass	$12.04kg$
Body Mass	$10.00kg$
Thigh Mass	$0.46kg$
Shin Mass	$0.31kg$
Foot Mass	$0.25kg$
Body's Moment of Inertia about COM	$0.100kgm^2$
Thigh's Moment of Inertia about COM	$0.0125kgm^2$
Shin's Moment of Inertia about COM	$0.00949kgm^2$
Foot's Moment of Inertia about COM	$0.00134kgm^2$
Thigh Length	$0.42m$
Shin Length	$0.42m$
Ankle Height	$0.04m$
Foot Length	$0.15m$

1.3 Objective and Scope

In terms of walking gaits, bipedal locomotion research can be classified into two main areas: 1) static walking and 2) dynamic walking. In static walking, the biped has to move very slowly so that its dynamics can be ignored. A static walking algorithm usually tries to maintain the biped's projected center of gravity (PCOG) within the boundary of the supporting foot or the smallest convex hull containing the two feet during the double support phase. By obeying this constraint and walking sufficiently slowly, the biped would not fall over if it freezes all its joints at any instance. Static walking is simple to implement because only geometry information is required to compute the location of the projected center of gravity. However, the walking speed and step size are limited.

In dynamic walking, the motion is faster and hence, the dynamics are not negligible. If the biped suddenly freezes all its joints, its momentum may cause it to topple over. In dynamic walking, one can look at the zero moment point (ZMP) [112] rather than PCOG. The ZMP is the point where the resulting reaction moment vector on the foot does not have a component in the horizontal plane (assuming the foot is on a horizontal ground). However, the ZMP does not have direct implications for walking stability. The choice of an appropriate and alternating sequence of steps between left and right leg results in walking stability. Dynamic walking usually results in larger step lengths and greater efficiency than static walking. However, the stability margin of dynamic walking is much harder to quantify.

The objective of this thesis is to synthesize and investigate a general control architecture for M2, or other robots that have the same DOF, to achieve 3D dynamic bipedal walking. In subsequent chapters, the walking task refers to the dynamic case unless otherwise specified. The control architecture is based on a divide-and-conquer approach in which the 3D bipedal walking task is decomposed into smaller subtasks. A learning method is applied to those

Table 1.2: Specifications of M2

Description	Value
Total Mass	$25.00kg$
Body Mass	$12.82kg$
Thigh Mass	$2.74kg$
Shin Mass	$2.69kg$
Foot Mass	$0.66kg$
Body's Principal Moment of Inertia	
x-axis	$0.230kgm^2$
y-axis	$0.230kgm^2$
z-axis	$0.206kgm^2$
Thigh's Principal Moment of Inertia	
x-axis	$0.0443kgm^2$
y-axis	$0.0443kgm^2$
z-axis	$0.00356kgm^2$
Shin's Principal Moment of Inertia	
x-axis	$0.0542kgm^2$
y-axis	$0.0542kgm^2$
z-axis	$0.00346kgm^2$
Foot's Principal Moment of Inertia	
x-axis	$0.000363kgm^2$
y-axis	$0.00152kgm^2$
z-axis	$0.00170kgm^2$
Hip Spacing	$0.184m$
Thigh Length	$0.432m$
Shin Length	$0.432m$
Ankle Height (from ankle's pitch axis)	$0.0764m$
Foot Length	$0.20m$
Foot Width	$0.089m$

subtasks that do not have simple solutions. Besides being able to achieve a stable walking gait, the resulting algorithm should achieve the following walking specifications:

1. The biped should be able to walk even when the step length is constrained.
2. The biped should be able to vary the walking speed.

The control algorithm should also have the following characteristics:

1. It should be easy to implement.
2. It should be applicable for real-time implementation.
3. It should be applicable to bipeds of different mass and length parameters.
4. The learning rate of the learning component to achieve the learning target should be high.

These specifications will be discussed in more detail in Section 3.2. The scope of this research is restricted to bipedal walking on level ground along a linear path. The analysis will be carried out in a dynamic simulation environment. Only the transient behavior from standing to forward walking is included in the analysis. The transient from walking to standing will not be studied though it should be achievable by the proposed approach. This thesis will also not study other behaviors of bipeds like static walking, turning, jumping, running, etc.

1.4 Approach

This section briefly explains how the specifications presented in the previous section can be achieved or implemented. The key philosophy adopted in this thesis is to seek the simplest possible control algorithm that satisfies the specifications stated in Section 1.3. The reason for such a guideline is that bipedal walking is usually very complex (especially dynamic walking). Therefore, it is not advisable to adopt very complex control algorithms. Otherwise, the overall system is most likely to be extremely complex or even intractable.

One of the ways to reduce the complexity of the problem is by task decomposition or divide-and-conquer approach. For example, 3D bipedal walking can be broken down into motion controls in the transverse, sagittal and frontal planes (see Figure 1-3). Each of these can then be considered individually.

Pratt et al. [80] proposed an intuitive control algorithm called "Turkey Walking" for planar bipedal walking based on a control language called Virtual Model Control. In the algorithm, the planar walking task is decomposed into three subtasks: 1) to maintain body height; 2) to maintain body speed; and 3) to maintain body posture.

The algorithm is well behaved and can easily be extended for rough terrain locomotion [20]. However, the control mechanism for the body speed can only be applied during the double support phase. It requires the double support phase to be significant in order to do the job. This requires the swing leg to be swung rapidly so as to produce longer period of the double support phase. This is hard to achieve when the walking speed is high because the real robot usually has limited joint torques. Furthermore, fast swing-leg motion consumes

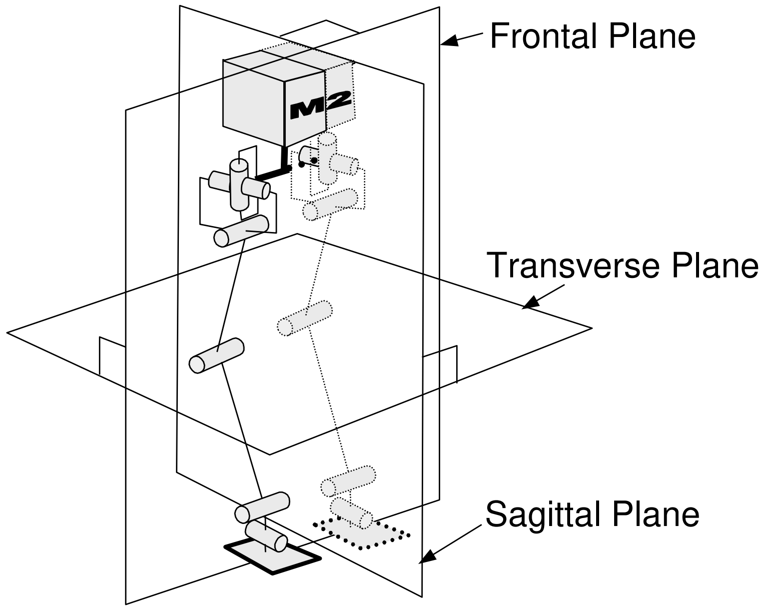


Figure 1-3: The orthogonal planes: Sagittal, Frontal and Transverse.

more energy. Due to the deficiency of the speed control based on the double support phase, this thesis does not adopt this strategy. Instead, the swing leg behavior is considered to be the key determinant for the gait stability. This idea is applicable to both forward speed stabilization and lateral balance of bipeds. However, due to the complexity of the bipedal walking sytem, there is no closed form expression for the foot placement parameters in terms of the desired walking speed or other gait parameters. As such, a learning approach is proposed for the swing leg task.

In this thesis, the divide-and-conquer approach used in the "Turkey Walking" algorithm is extended to the 3D walking task. In the 3D walking task, the motion control is first partitioned into the three orthogonal planes. Each of these is further subdivided into smaller subtasks. A learning algorithm is used to assist those subtasks that are not easily accomplished by simple strategies or control laws. The overall control algorithm should satisfy the desired performance specifications discussed before.

In this approach, the learning time is expected to be shorter than those learning approaches that learn all the joint trajectories from scratch. This is because the scope of learning is smaller and less complex if the learning algorithm focuses on just a subtask. Furthermore, the previous research results of the Turkey Walking algorithm can be utilized. In summary, the robot should only learn the tasks or parameters that really require learning. A detailed description of this approach is presented in Chapter 3.

1.5 Simulation Tool

SD FAST, a commercial dynamic simulation program from Symbolic Dynamics, Inc [91] is used to generate the dynamics of the simulated bipeds. It is based on Newtonian mechanics for interconnected rigid bodies. An in-house simulation package called the Creature Library [88] is used to generate all the peripheral C routines (except for the dynamics) for the simulation after a high level specification of the biped is given.

The dynamic interaction between the biped and the terrain is established by specifying four ground contact points (two at the heel and two at the toe) beneath each of the feet. The ground contacts are modeled using three orthogonal spring-damper pairs. If a contact point is below the terrain surface, the contact model will be activated and appropriate contact force will be generated based on the parameters and the currect deflection of the ground contact model. If a contact point is above the terrain surface, the contact force is zero.

Before a simulation is run, the user needs to add a control algorithm to one of the C programs. In the control algorithm, only information that is available to the physical robot is used. The body orientation in terms of the roll, pitch, yaw angles and the respective angular velocities are assumed to be directly available. All the joint angles and angular velocities are also known. The contact points at the foot provide information about whether they are in contact with the ground or not.

The output of the control algorithm is all the desired joint torques. The dynamics of the actuators are ignored in the simulation. That is, the actuators are considered to be perfect torque or force sources.

1.6 Thesis Contributions

The contributions of this thesis are:

1. The synthesis of a general control architecture for 3D dynamic bipedal walking based on a divide-and-conquer approach. Learning is applied to achieve gait stability. No prior mathematical model is needed.

2. The demonstration that dynamic walking can be achieved by merely learning parameters of swing leg control.

3. The design and investigation of several walking algorithms based on the proposed control architecture.

4. The demonstration of the effects of local control mechanism on learning rate and stabilization.

5. The application of the notion of symmetry for lateral balance during forward walking. A quantitative analysis has been done to verify the approach.

6. The demonstration of the generality of the control architecture by applying the algorithm to different bipeds that have similar degrees of freedom but different inertia and length parameters.

1.7 Thesis Outline

Chapter two gives a literature review of the bipedal locomotion research that is relevant to this thesis. It groups bipedal walking research into four categories. Examples of each of the categories are discussed.

Chapter three presents the common characteristics of bipedal walking. It also describes the desired performance of the walking task and the control algorithm. The motivation for the approach adopted in this thesis is then discussed. A control architecture for the 3D dynamic walking task is presented.

Chapter four describes the methods or tools utilized to formulate the walking control algorithm based on the proposed control architecture. It includes Virtual Model Control which is used to facilitate the transformation of the desired behavior of the stance and swing legs into respective joint torques. It also describes a reinforcement learning method called Q-learning that is used to learn the key swing leg parameters.

Chapter five first presents an analysis of the sagittal plane walking motion. The swing leg dynamics is demonstrated to be significant and not ignorable in general based on a simple double pendulum model. The application of learning to the control algorithm is also justified. After that, several algorithms for the sagittal plane walking are presented. The usage of the local control at the stance ankle joint in speeding up the learning process is illustrated by simulations. The generality of the proposed control architecture is also demonstrated by applying one of the algorithms to two different bipeds.

Chapter six first presents some heuristics of the frontal plane motion. Then, the frontal and transverse planes algorithms are presented. They are combined with the sagittal plane to form a 3D dynamic walking algorithm. Two approaches for the lateral balance are highlighted. One is based on the learning approach and the other is based on the notion of symmetry. A simple model is constructed to verify the latter approach. All the algorithms are verified in the simulation environment using the simulated M2.

Chapter seven includes two other implementations to illustrate the flexibility and generality of the proposed control architecture. One of these is the variable walking speed implementation. The other is to achieve a different walking posture.

Chapter eight concludes the work of this thesis and suggests future work.

Chapter 2

Literature Review

Much research has been done in the area of bipedal locomotion in recent years. Since bipedal locomotion is a complex problem and the scope of bipedal locomotion is wide, most researchers have restricted their research to a smaller area. For example, some researchers have concentrated on the area of mechanics, some on control, some on energetics of biped locomotion, etc. Some researchers further simplify matters by partitioning the biped gait and restricting their analysis either to the sagittal plane or the frontal plane.

For a biped robot to achieve a walking gait, the control algorithm needs to comply with the constraints of the bipedal system. One important constraint is the unpowered DOF between the foot and the ground [112]. This constraint limits the trajectory tracking approach used commonly in manipulator research and is one of the main reasons bipedal locomotion is such a challenging research area.

This chapter classifies the control approaches adopted for dynamic bipedal walking into four categories: 1) model-based; 2) biologically inspired; 3) learning; and 4) divide-and-conquer. Various approaches for each of the categories will be presented. These classes of approach discussed in this chaper are by no means mutually exclusive or complete. Some research may combine several of these approaches when designing a control algorithm for the bipedal walking.

2.1 Model-based

In this approach, a mathematical model of the biped derived from physics is used for the control algorithm synthesis. There is a spectrum of complexity for a biped model, ranging from a simple massless leg model to a full model which includes all the inertial terms, friction, link flexibility, actuator dynamics, etc. Except for certain massless leg models, most biped models are nonlinear and do not have analytical solutions. In addition, there is underactuation between the stance foot and the ground; and unknown discrete changes in the state due to foot impact.

The massless leg model is the simplest model for characterizing the behaviors of the biped. The body of the biped is usually assumed to be a point mass and can be viewed to be an inverted pendulum with discrete changes in the support point. The massless leg model is applicable mainly to a biped that has small leg inertia with respect to the body.

In this case, the swing leg dynamics can be ignored under low walking speed.

Kajita et al. [52, 50] have derived a massless leg model for a planar biped that follows a linear motion. During single support phase, the resulting motion of the model is like a point mass inverted pendulum with variable length. The dynamic equations of the resulting linear motion can be solved analytically. From the model, the touchdown condition of the swing foot is determined based on an energy parameter. Inverse kinematics is used to specify the desired joint trajectories and simple control law is used at each joint for tracking.

When the leg inertia is not ignorable, it needs to be included in the biped model. One basic model that includes the leg inertia is the Acrobot model [118, 73]. It is a double pendulum model with no actuation between the ground and the base link (corresponding to the stance leg). This is commonly used to characterize the single support motion of the biped.

When the leg inertia and other dynamics like that of the actuator, joint friction, etc are included, the overall dynamic equations can be very nonlinear and complex. A linearization approach is usually adopted to simplify these dynamic equations. The linearization can be done with respect to selected equilibrium points to yield sets of linearized equations.

Miura and Shimoyama [71] built a 3D walking biped that had three links and three actuated degrees of freedom: one at each of the hip roll joints and one for fore and aft motion of the legs. The ankle joints were limp. The biped was controlled using a linearized dynamic model. The model assumed that the motions about the roll, pitch and yaw axes were independent. It also assumed that the yaw motion was negligible and there was no slipping between the feet and the ground. After selecting a set of feasible trajectories for the joints, state feedback control laws were then formulated for the joints to generate compensating inputs for the nominal control inputs. The control laws ensured the convergence of the actual trajectories to the desired trajectories. Since they adopted a linearized model for the biped, the motion space had to be constrained to a smaller one so that the model was not violated.

Mita et al. [70] proposed a control method for a planar seven-link biped (CW-1) using a linear optimal state feedback regulator. The model of the biped was linearized about a commanded attitude. Linear state feedback control was used to stabilize the system based on a performance function so that the biped did not deviate from the commanded attitude. For the gait control, three commanded attitudes were given within a gait interval and the biped could walk arbitrary steps with one second period. They assumed that the biped had no torque limitation at the stance ankle. The biped had large feet so that this assumption was valid.

2.2 Biologically Inspired

Since none of the bipedal robots match biological bipeds in terms of mobility, adaptability, and stability, many researchers try to examine biological bipeds so as to extract certain algorithms (reverse engineering) that are applicable to the robots.

The steps taken are usually as follows. First, the behaviors of the biological system are observed and analyzed. Then, mathematical models and hypotheses for the behaviors are proposed. After that, they are verified through simulations or experiments. However, in most cases, the observations only constitute a small piece of the big puzzle. Thus, their applications to the bipedal robots can be limited.

This subsection includes two main research areas in this category. The first is the Central Pattern Generators (CPG) approach which is based on the findings that certain legged animals seem to centrally coordinate the muscles without using the brain. The second is the passive walking approach that is based on the observation that human beings do not need high muscle activities to walk.

2.2.1 Central Pattern Generators (CPG)

Grillner [34] found from experiments on cats that the spinal cord generated the required signal for the muscles to perform coordinated walking motion. The existence of a central pattern generator that is a network of neurons in the spinal cord was hypothesized.

The idea of CPG was adopted by several researchers for bipedal walking [103, 104, 7]. One typical approach is the use of a system of coupled nonlinear equations to generate signals for the joint trajectories of bipeds. The biped is expected to achieve a stable limit cycle walking pattern with these equations.

Taga [103, 104] applied a neural rhythm generator to a model of human locomotion. The neural rhythm generator was composed of neural oscillators (a system of coupled nonlinear equations) which received sensory information from the system and output motor signals to the system. Based on simulation analysis, it was found that a stable limit cycle of locomotion could emerge by the principle of *global entrainment*. The global entrainment was the result of the interaction among the musculo-skeletal system, the neural system and the ground.

Bay and Hemami [7] demonstrated that a system of coupled van der Pol oscillators could generate periodic signals. The periodic signals were then applied to the walking task to produce rhythmic locomotion. In their analysis, the dynamics of the biped such as the force interactions between the support leg(s) and the ground were not considered.

One disadvantage of a CPG approach based on a set of coupled nonlinear equations is that there are many parameters to be tuned. It is also difficult to find a set of parameters that enable entrainment of the overall system. Even if a periodic walking behavior can be obtained, it is hard to predict the walking behavior when it is subjected to disturbance due to uncertain ground profile. It is also hard to change the behavior or add more behaviors.

2.2.2 Passive Dynamics

Passive dynamics study provides interesting natural dynamic model for the mechanics of human walking [63, 64, 30]. It was partly inspired by a bipedal toy that was capable of walking down a slope without any actuator system. The toy rocked left and right in a periodic motion. When a leg is lifted in the process, it could swing freely forward and arrived in a forward position to support the toy for the next half period of the motion. If the slope is within a certain range, a stable limit cycle behavior can be observed. When this occurs, the work done on the toy by the gravitational force equals the energy loss by the biped (e.g., impact loss, friction at the joints between the body and the legs).

Most passive dynamics analyses study the stability and periodicity issue of passive walking. Similar to the bipedal toy, most passive walking implementations do not incorporate any actuator systems or active controllers in the walker. The walking behaviors are usually demonstrated on a downslope so that the gravitational potential energy can be utilized to generate the walking motion.

Though passive walking has nice properties like being able to achieve a minimum energy gait without active control, it is sensitive to parameter variations [63], such as, the mass distribution, joint friction etc. During physical implementation, iterative tuning is usually required before a successful implementation can be achieved. Furthermore, it is still unclear how it can be implemented for rough terrain walking.

2.3 Learning

The learning approach is often touted to possess biological behaviors. It is evident from observing toddlers who are just starting to walk that walking is a learned process. Learning is also commonly applied to systems whose models are hard to derive or implement. In some cases, learning is used to modify a nominal behaviors that are generated based on a simplified model.

Miller, Latham and Scalera [69] adopted a simple gait oscillator which generates the trajectory of the swing leg for a simulated planar biped. The input to the oscillator was the step time and the desired step length based on a fixed model. A CMAC (Cerebellar Model Articulation Controller) network was used to adjust the desired step length based on past experience. The purpose was to achieve a desired forward velocity at the end of each step. The CMAC network was trained under two different times scales. Based on the velocity error at the end of each step, a supervised learning approach was adopted to adjust the step length prediction of the last control cycle of the previous step. Also, the step length prediction for each control cycle was adjusted based on the prediction in the next control cycle. When the biped fell forward or backward during a step, the CMAC network step length prediction at the end of the last step was adjusted by a fixed value in a direction opposite to the fall direction. The control algorithm was successfully implemented for a simulated planar three-link biped with massless legs.

Wang, Lee, and Gruver [113] presented a neuromorphic architecture based on supervised learning for the control of a three-link planar biped robot. Based on the biped's dynamic model, a control law was obtained by nonlinear feedback decoupling and optimal tracking approach. Then, they used a hierarchical structure of neural networks to learn the control law when the latter was controlling the biped. They assumed that the desired joint trajectories were available. However, in physical implementation, due to the torque limitation of the stance ankle, it is difficult to obtain a working set of joint trajectories without violating this constraint.

Miller [66] designed a learning system using neural networks for a biped that was capable of learning the balance for side-to-side and front-to-back motion. With the help of several control strategies, the neural networks learn to provide feedforward control to the joints. The biped had a slow walking gait after training.

Benbrahim and Franklin [9] applied reinforcement learning for a planar biped to achieve dynamic walking. They adopted a "melting pot" and modular approach in which a central controller used the experience of other peripheral controllers to learn an average control policy. The central controller was pre-trained to provide nominal trajectories to the joints.

Peripheral controllers helped the central controller to adapt to any discrepancy during the actual run. The central controller and some of the peripheral controllers used simple neural networks for information storage. No dynamic model of the system was required in the implementation. One disadvantage of the approach was that the nominal joint trajectories were required to train the central controller before any learning began. This information is usually not easily obtainable.

In summary, there are many ways to implement a learning algorithm for the biped. It depends not only on the level of the control hierarchy to which learning is applied but also on how the control task is decomposed. Of course, the design of the bipedal robot also greatly influences the algorithm, especially the actuator and sensor systems and the degrees of freedom. Furthermore, just as in any optimization techniques, learning success relies very much on the performance index chosen.

2.4 Divide-and-conquer

Due to the complexity of the bipedal walking system, many implementations break the problem into smaller sub-problems that can be solved more easily. Intuition of the problem is usually required for such an approach, both in deciding how to break down the problem and how to solve the smaller sub-problems. Intuition can be obtained by observing the behavior of bipedal animals (similar to the biologically inspired approach), by analyzing simple dynamic models, etc.

Raibert's control algorithms [86] for hopping and running machines mostly utilized the divide-and-conquer approach. For example, the control algorithm for a planar one-legged hopping machine was decomposed into: 1) the hopping motion (vertical); 2) forward motion (horizontal) and; 3) body posture. These sub-tasks were considered separately and each was implemented using simple algorithm. This resulted in a simple set of algorithms for the hopping task.

Pratt et al. [81, 80] presented a control algorithm (called "Turkey Walking") based on a divide-and-conquer approach for planar bipedal walking task of a biped called "Spring Turkey". The walking cycle was first partitioned into two main phases: double support and single support. A simple finite state machine was used to keep track of the currently active phase. In the double support phase, the task of the controller consisted of three sub-tasks: 1) body pitch control; 2) height control and; 3) forward speed control. In the single support phase, the task of the controller consisted of two sub-tasks: 1) body pitch control and 2) height control. Based on this framework, a control language called Virtual Model Control (VMC) was used to design the control algorithm. The resulting algorithm was simple and low in computation since no dynamic equations were used. It was successfully implemented to control a planar biped for dynamic walking on level ground in real-time. Chew [18] has extended the algorithm for rough terrain locomotion.

The divide-and-conquer approach has been proven to be simple and effective for practical implementations. However, not all the sub-tasks contain direct or analytic solutions. Some may require tedious iterative parameter tunings.

2.5 Summary

This chapter gives a classification of the research on dynamic bipedal walking. It is by no means complete. Many excellent studies were omitted to keep this chapter concise. In theory, the model-based approach may be an excellent approach if one could find a model that is not too complex to compute and that is not too simple to miss all the nonlinear behaviors. One disadvantage is that a general model is usually too complex to analyze and implement.

The Central Pattern Generator (CPG) approach is excellent in generating periodic gaits. However, CPG is usually achieved by a set of coupled nonlinear equations which do not have analytical solutions. The overall system may become intractable or hard to analyze.

The passive walking study is great for demonstrating why human beings do not consume high energy during normal walking. However, most studies only look at the behaviors of the passive walking biped when it walks down a slope. If the idea is to be applied to a bipedal robot, it is necessary to find a way to extend the approach for walking over different terrains rather than just downslope.

The learning approach seems to be very promising. However, intuition is still required to decide: 1) to which level of the control algorithm learning should be applied; 2) the number of learning agents to be installed; 3) which performance measure should be adopted, etc. Learning approaches may become intractable if there are too many learning agents (especially if they are coupled to one another). It may also be intractable if the bipedal walking task is not broken down into small sub-tasks. In this case, it is very difficult to design a performance measure.

The divide-and-conquer approach has been demonstrated to be effective in breaking the complex walking task into smaller and manageable sub-tasks. Some of the sub-tasks can then be easily solved by simple control laws. However, some sub-tasks do not have simple or analytical solutions. Even if control laws can be constructed for these sub-tasks, the control laws usually require iterative tuning of the parameters.

The next chapter presents a control architecture which is synthesized using both the divide-and-conquer approach and the learning approach. Here, the learning approach plays a supplementary role in the algorithm. It is applied only when a subtask does not have a simple solution.

Chapter 3

Proposed Control Architecture

This chapter first presents the general characteristics of bipedal walking. Next, the desired performance of the dynamic bipedal walking task and the control algorithm are specified. A previous algorithm for planar bipedal walking task is then examined. Motivated by the deficiency of this algorithm, a control architecture with learning as one of the components is proposed. The proposed control architecture is based on a divide-and-conquer framework. Using this framework, the three-dimensional walking task can be broken down into small subtasks. Learning methods are applied to the subtasks on an as-needed basis.

3.1 Characteristics of Bipedal Walking

This section presents the general characteristics of bipedal walking. Bipedal walking can be viewed to be a sequence of alternating swing leg executions that results in the translation of the body in spatial coordinates. A right alternating motion of the legs is important to ensure stable gaits. Bipedal walking is different from running in that there is at least one leg on the ground at anytime. The stance leg provides the support for the body whereas the swing leg is swung forwards to the next support location. There exists a support exchange period where both feet are on the ground. After this period, the roles of the legs are switched.

Within each step of forward walking, the body velocity in the sagittal plane first slows down and then speeds up. This behavior is similar to that of an inverted pendulum. It also sways from side to side in the frontal plane as the biped progresses forwards. Symmetry and cyclic behaviors usually exist for steady walking on level ground. When rough terrain is encountered, the priority is to adjust the walking posture and to constantly try to avoid bad foothold positions that may lead to a falling event.

Most biological bipeds possess a continuum of stable walking patterns. This is partly attributed to the redundant degrees of freedom. It seems that the bipeds can easily choose a pattern from the continuum set based on some optimization critieria, for example, minimum energy, overall walking speed, etc.

3.2 Desired Performance of Bipedal Walking Task and Control Algorithm

This section describes the desired performance of the bipedal walking task and the control algorithm. First, the desired performance of the task is discussed. For a bipedal robot to justify its existence, it should be able to achieve bipedal walking that has the following basic capabilities:

- able to achieve variable walking speed: This feature is a standard requirement for any locomotion system. It includes the capability to handle the transient phase too.
- able to control step length: This feature is needed for walking over rough terrain that restricts the foot placement locations. For example, in a stair walking task, it is important to place the swing foot within the boundary of the next elevation while also maintaining gait stability.
- able to adapt to ground variation: This feature is important for walking over unstructured terrain. The ground elevation may vary from step to step. Thus, the biped needs to adjust its motion accordingly so that a stable walking gait can be maintained.
- able to change walking direction: This feature is an inherent requirement for locomotion. Without this, the locomotion capability is limited.

Note that this list does not include other behaviors of the biped. For example, it does not include stopping, static balancing, etc. This thesis only explores the first two capabilities and designs control algorithms that can achieve them.

Besides being able to achieve the desired performance for the walking task, a control algorithm should possess the following characteristics if it is to be applied to a physical system. It should be:

- easy to implement: The use of full dynamic equations (to include all the degrees of freedom and all the rigid body inertias) should be avoided since the equations are too complex to be implemented. Also, parameter tuning will be a problem if the dynamic equations are too complex.
- suitable for real-time application: The computation requirement should be low. The learning rate should not be too slow.
- general: The algorithm should be applicable across bipeds with different actuator characteristics and; inertia and length parameters. This requirement is not a must, but such generality is useful if the algorithm needs to be applied to bipeds that have different sizes.

3.3 Divide-and-conquer Approach for 3D Dynamic Bipedal Walking

A 3D bipedal walking system can be very complex to analyze if it is not partitioned into smaller components or sub-tasks. It is difficult to apply a unified control algorithm for such a complex system. If there exists such an unified control algorithm, its complexity may be very high and thus hard to analyze. For example, a multi-layer neural network may be a possible candidate for such an algorithm. The inputs may consist of all the state variables and desired behaviors. The outputs may consist of all the desired actuator behaviors. However, the key problem is that an input/output example set needed for training is usually unavailable a priori. Even if the training set is available, finding an effective evaluation function is difficult. The complexity of the algorithm may also be so high that it takes a long time to train the network.

In this thesis, a divide-and-conquer approach is adopted in the formulation of the dynamic walking algorithm. By studying each subtask separately, the problem becomes less complex. Appropriate control algorithms can then be applied for each of them. This section presents the framework for such a task decomposition approach.

It is assumed that the motions in the three orthogonal planes (sagittal, frontal and transverse, see Figure 1-3) can be independently considered. They are separately discussed in the following subsections.

3.3.1 Sagittal Plane

Sagittal plane motion is usually the largest during normal forward walking. The task in this plane can be further decomposed as in Pratt et al. [81, 80] into the following subtasks: 1) to maintain body pitch; 2) to maintain body height; and 3) to maintain desired walking speed. It is very easy to achieve the first and second subtasks in the sagittal plane. However, the third subtask is much more complex. It is directly associated with gait stabilization.

There are a number of speed control mechanisms in the sagittal plane. They can be partitioned into two main classes:

- global control: This refers to any speed control mechanism that can achieve global speed control by itself.
- local control: This refers to any speed control mechanism that can only achieve limited or local speed control.

Examples of local speed control mechanisms are those based on the double support phase [80, 20], the biped's torso (using torso as a limited reaction wheel), the stance ankle, etc. Pratt et al. [80, 20] have used the double support phase for the walking speed regulation in the sagittal plane. It is very simple to implement. However, the duty factor of the double support phase needs to be significant in order to be effective. Such a requirement is hard to fulfil when the biped is walking at high speed. Such an approach is also not energy efficient and demands high actuator bandwidth since the swing leg needs to be swung quickly to allow the double support phase to have a large duration. In three dimensional walking, the double support based speed control is more tricky since there may be complex coupling between the frontal and the sagittal plane control. Due to these limitations, this mechanism is not adopted in this thesis.

Another local speed control mechanism is based on the stance ankle. Pratt and Pratt [82] proposed a "virtual toe point" approach which is one of the ways to implement stance ankle based control. In subsequent chapters, it will be demonstrated how stance ankle control can be used to supplement the control algorithm constructed in this thesis.

Other local speed control mechanisms include strategy based approaches. One example is the addition and removal of energy at strategic times [83]. Strategy based approaches can be powerful, but the implementations usually require intuition and trial-and-error tunings.

In constrast, there is only one mechanism for the *global* speed control. It is based on swing leg control. If swing leg control is badly executed, it is possible that no other speed control mechanism can prevent the biped from falling.

Swing leg control, therefore, is the key to walking speed control because it affects the global walking velocity. A swing leg behavior is determined by the following approaches: 1) by prescribing the swing leg trajectory in the joint or Cartesian space and/or 2) by exploiting natural dynamics [84]. The question is how to execute the swing leg in the sagittal plane such that stable walking can be achieved in the same plane.

3.3.2 Frontal Plane

For normal walking along a straight path, frontal plane motion is smaller than sagittal plane motion. In the frontal plane, the dynamic walking task can be decomposed into: 1) maintaining the body roll and 2) maintaining lateral balance.

As in the sagittal plane, body posture control can be achieved easily. However, lateral balance is not as simple. The control mechanisms for it can also be partitioned into global control and local control. The local control mechanisms include those that utilize the double support phase, the biped's torso, the stance ankle (roll) etc. The thesis only explores the usage of the stance ankle in the control algorithm, especially how it can be used to compliment the global control mechanism. The global control mechanism again consists of only the one that is based on the swing leg control. Thus, it is also the key mechanism for lateral balance.

3.3.3 Transverse Plane

For normal walking along a straight path, transverse plane motion is usually the simplest to control. In this case, the walking task can be decomposed into: 1) maintaining the body yaw angle and 2) maintaining the swing foot yaw angle. Both the body yaw angle and the swing foot yaw angle can be set to zero (i.e. facing forwards). Both subtasks can be easily achieved. Transverse plane motion is important and more complex for tasks like turning, especially while the biped is walking dynamically.

3.4 Swing Leg Behavior

The previous section has identified swing leg control to be the key gait stabilization mechanism for dynamic bipedal walking. Although swing leg motion is an important determinant for the stabilization of the walking speed and the lateral motion, it is difficult to obtain a closed form expression for it. This is because the dynamic equations of a biped are complex and nonlinear. This is further complicated by the unpowered DOF between the stance foot and the ground during the single support phase [112]; and the discontinuities caused by the impact at the support exchange. The overall motion of the system is usually strongly coupled to swing leg motion unless the leg's inertia is negligible or the motion is slow.

Some research assumes that the effect of the swing leg is negligible. Then, the resulting system can be represented by an inverted pendulum. However, this assumption becomes invalid if the mass of the leg is significant. Furthermore, for fast walking, the dynamics of the swing leg becomes significant. The swing leg dynamics may also be utilized for walking, for example, to create reaction torque at the hip. If its effect is ignored, no such utilization is possible. The configuration of the swing leg also affects the projected center of gravity of the biped, thus changing the gravitational moment about the supporting leg.

The importance of the swing leg behavior prompted several biomechanics researchers to perform in depth studies of it, especially relating to human walking. Redfern and Schumann [87] analyzed the swing trajectories of the foot during gait with respect to the pelvis. They proposed a model of the foot placement control for stable base support of human locomotion. Experimental data were collected to test the model during walking trials of different speeds. Townsend [110] found that lateral stability of human walking is maintained through foot placements based on conditions at each new step. The body center of mass trajectories can be controlled by foot placement alone. Beckett and Chang [8] investigated the behavior of the swing leg during normal walking. They also studied the energy consumption of walking and concluded that there is a natural gait at which a person can walk with minimum effort.

The research works listed above were carried out for human beings. It is not clear whether the model derived for human walking can be generally applied to other bipeds or used to control a bipedal robot. A bipedal robot usually has different inertia and length parameters compared to human beings. Besides, both have different actuator systems. Even for human walking, individuals may walk differently depending on their heights, types of shoe, moods, etc. Thus, the usefulness of the anthropometric data for controlling a bipedal robot is questionable.

3.5 Proposed Control Architecture

In this section, a general control architecture is synthesized for dynamic bipedal walking based on the divide-and-conquer framework discussed earlier. As mentioned before, the dynamic walking task can be partitioned into the three orthogonal planes: frontal, sagittal and transverse. Learning is mainly used to learn the swing leg behavior in the sagittal plane and the frontal plane. The biped has no nominal parameters for the swing leg to start with. The learning agents will select appropriate swing leg parameters, for example, step length, swing time, etc, and evaluate them accordingly. The learning is done online and the evaluation is done discretely at, for example, the support exchange event. The advantage of using the learning approach is that non-linearity due to bandwidth limitation, torque limitation, etc is encapsulated from the perspective of the controller or learner. The overall control architecture can be summarized as shown in Figure 3-1.

Based on this architecture, a control algorithm can be formulated for the three-dimensional dynamic bipedal walking task. In particular, *Virtual Model Control* (VMC) [80, 20] and a learning tool called *Q-learning* which belongs to the class of reinforcement learning (RL) methods [45] are used to formulate walking algorithms that satisfy the desired performance described in Section 3.2.

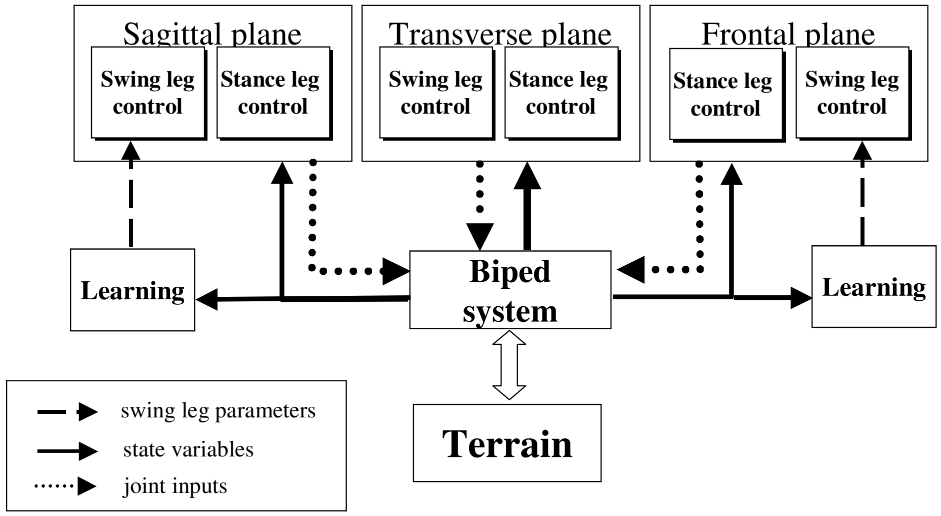


Figure 3-1: Proposed control architecture

Chapter 5 demonstrates several implementations for the sagittal plane motion. Chapter 6 includes the frontal plane algorithm and the transverse plane algorithm. It also describes how the algorithms for each of the three orthogonal plane motions are assembled to yield a three-dimensional control algorithm. Before that, the next chapter describes the tools used in the algorithms.

Chapter 4

Implementation Tools

This chapter describes the tools that are used in the control algorithm implementations. A control language for legged locomotion called Virtual Model Control (VMC) [81, 80] is first presented. It is used to facilitate the implementation of the divide-and-conquer approach described in the previous chapter. Control algorithms for the stance leg and swing leg are designed based on VMC. The algorithms generate the desired torque for all the joints. However, they do not necessarily result in stable motion. Learning is required to select suitable swing leg parameters that result in a stable walking behavior. This chapter describes a learning approach called reinforcement learning. A reinforcement learning algorithm called Q-learning [114] is presented. Then, a function approximator called Cerebellar Model Articulation Controller (CMAC) [3] is described. It is a component of the Q-learning algorithm that uses function approximator.

4.1 Virtual Model Control

Virtual Model Control (VMC) [81, 80] is a control language for legged locomotion. It allows us to work in a more intuitive space (e.g., Cartesian space) instead of a space (e.g. joint space or actuator space) that suits the robot. This section describes the Virtual Model Control approach and how it can be applied to bipedal walking.

4.1.1 Description of Virtual Model Control (VMC)

In Virtual Model Control (VMC), a control algorithm is constructed by applying virtual components (which usually have corresponding physical parts, for example, spring, damper, etc) at some strategic locations of the biped. The virtual components interact with the biped and generate a set of virtual forces based on their constitutive equations. Then these forces (represented by a vector $\vec{f}$) can be transformed into the joint space (represented by a vector $\vec{\tau}$) by a Jacobian matrix ${}^A_B J$:

$$\vec{\tau} = {}^A_B J^T \, {}^A_B \vec{f} \tag{4.1}$$

where ${}^{A}_{B}J$ is the Jacobian matrix which transforms the differential variation in the joint space into the differential variation of reference frame B with respect to reference frame A. The superscript ^{T} denotes the transpose of a matrix. In Virtual Model Control, frame B and frame A correspond to the action frame and reaction frame, respectively [81]. This expression is also commonly used in robotics literature to relate a force vector in the Cartesian space to an equivalent torque vector in the joint space [5].

Equation 4.1 requires much lower computation compared to the equations encountered in inverse kinematics and dynamics. It is also well-behaved since, for a given force vector in the virtual component space, a corresponding torque vector can always be obtained in the joint space even if the robot is in a singular configuration.

By changing the set points of the virtual components, an inequilibrium can be created in the system. The system will then adjust itself so that a new equilibrium position is reached. The biped will behave dynamically as if the actual physical components are attached to it provided that the actuators are perfect torque sources.

There is lots of freedom in the virtual components selection and placement. Some may result in simple implementations that are easy to visualize, whereas others may result in complex systems. The next subsection provides an example of how VMC can be applied to bipedal walking control.

4.1.2 Application to Bipedal Walking

When applying VMC to bipedal walking, the placement of virtual components requires intuition. There are more than one set of virtual components which can achieve a given task. Let's consider Spring Flamingo (Figure 1-1), which is constrained to move in the sagittal plane. By adopting the divide-and-conquer approach as described in Subsection 3.3.1, one possible set of virtual components is shown in Figure 4-1 [81]. A virtual spring-damper is attached to the hip along the vertical direction to generate a virtual vertical force f_z at the hip. It is used to control the vertical height of the biped. A virtual damper is attached to the hip along the horizontal direction to generate a virtual horizontal force f_x at the hip. This is used to control the horizontal velocity of the biped. A rotational spring-damper is attached at the hip to generate a virtual moment m_α about the hip. This is used to maintain the upright posture of the biped's torso.

For the sagittal plane walking control, the inputs are the desired horizontal velocity $\dot{x}^d$, the desired vertical height of the hip z^d and the desired pitch angle α^d of the body. All these inputs define the desired motion of the body frame $O_bX_bZ_b$ with respect to the inertia frame OXZ. Assuming that the virtual components are linear, the following virtual forces/moment can be expressed as follows:

$$f_z = k_z(z_d - z) + b_z(\dot{z}_d - \dot{z}) \tag{4.2}$$

$$f_x = b_x(\dot{x}_d - \dot{x}) \tag{4.3}$$

$$m_\alpha = k_\alpha(\alpha_d - \alpha) + b_\alpha(\dot{\alpha}_d - \dot{\alpha}) \tag{4.4}$$

where k_i and b_i are the the spring stiffness and the damping coefficient, respectively, for the virtual components in i ($= x$, z or α) coordinate.

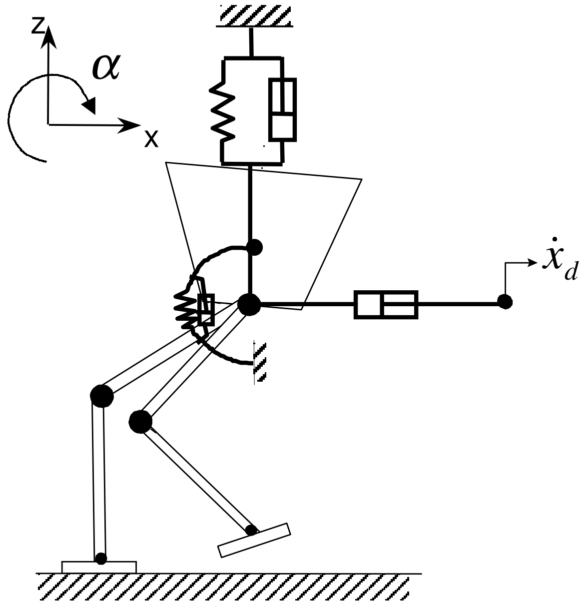


Figure 4-1: A possible set of virtual components to achieve walking control for *Spring Flamingo*

However, this set of virtual components can only be applied to the double support phase and not to the single support phase. During the single support phase, the biped loses one important actuated degree of freedom (assuming the stance ankle is limp). In this case, one of the virtual components cannot be realized. For example, the vertical and rotational spring-dampers can be implemented and the gravity can dictate the horizontal velocity.

In each phase, after the virtual force(s)/moment are computed, they are transformed into the joint space using the corresponding Jacobian matrices [81] (the Jacobian matrix for the single support phase is included in Appendix A). A state machine detects whether the biped is in the single support phase or in the double support phase. It is used to switch between the two different sets of virtual components. It also keeps track of whether a leg is the stance leg or swing leg. In this way, VMC enables the biped to walk without computing the inverse dynamics and the inverse kinematics. This significantly simplifies the computation. Hence, it is applicable to real-time implementation.

4.2 Reinforcement Learning

This section describes a learning paradigm called the reinforcement learning methods [45, 102, 10, 6]. Reinforcement learning is a class of learning problem in which a learner (agent) learns to achieve a goal through *trial-and-error* interactions with the environment. The learning agent learns only from reward information, and it is not told how to achieve the task. From failure experience, the learning agent reinforces its knowledge so that success can be attained in future trials. This learning approach is similar to how an animal learns a new skill.

Reinforcement learning is different from supervised learning in several ways. For the former problem, once a goal is set, the agent simply learns how to achieve the goal by trial-and-error interactions with its environment. The learning is done with the help of a critic rather than a teacher [102]. No explicit training data set (input/output pairs) is used in the training as in the supervised learning. This approach is useful for problems where there are no sample input-output pairs that could be used to train a function approximator like neural networks. Hence, the system is not required to work successfully using other conventional control approaches prior to learning. Reinforcement learning is also applicable to delayed reward problems where the effect of an action is known only after several time steps.

Most reinforcement learning analyses assume that the learning problems can be posed as Markov decision processes (MDPs) with finite state and finite action sets. MDP is a discrete-time dynamic system in which the state transition only depends on the present state i and the action u [1] taken. That is, if an action u ($\in U(i)$) is chosen in state i, the system will go to the next state j with probability $Pr(j|i,u)$ (also called the *state transition probability*). For each transition, say from i to j due to the action u, an immediate reward $r(i,u,j)$ is incurred.

A MDP problem can be classified into either an *finite horizon problem* or an *infinite horizon problem* [10]. In the former case, there is a finite number of stages, whereas there is infinite number of stages in the latter case. In this thesis, the bipedal walking task is posed as a discounted problem. This is a class of the infinite horizon problem which aims to maximize the discounted return R_t associated to t th stage:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{4.5}$$

where r is the reward received, $\gamma \in [0,1)$ is called the discount factor which indicates how much emphasis is put on the future reward, and the subscript denotes the stage number.

The action u is selected based on a policy π which is a function of the present state i. The aim of reinforcement learning techniques is to learn a policy that can achieve a given goal (e.g., mimimal expected cumulative cost). When applied to control problem, the agent indirectly learns a policy by estimating a value function called *Q-factors* that is a function of the state and the action [102]. For MDPs, the Q-factor of a state-action pair under policy π ($Q^\pi(i,u)$) represents the expected return R_t when the action u is taken at the state i, thereafter, following the policy π. For a discounted problem, $Q^\pi(i,u)$ at t th stage can be expressed as follows:

$$Q^\pi(i,u) = E^\pi[R_t|i_t = i, u_t = u] = E^\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}|i_t = i, u_t = u] \tag{4.6}$$

[1]The action u may be low level like the control input to a system or high level like decision making.

where $\gamma \in [0, 1)$ is a discount factor, r_t denotes the immediate reward received after the $(t-1)$th stage.

In most problems, the aim is to obtain optimal Q-factors $Q^*(i, u)$ which are defined as follows:

$$Q^*(i, u) = \max_{\pi} Q^{\pi}(i, u),\ \forall i \in I, u \in U(i) \tag{4.7}$$

$Q^*(i, u)$ gives the expected return when the agent takes the action u in the state i and adopts an optimal policy π^* thereafter. Based on $Q^*(i, u)$, an optimal policy π^* can easily be derived by simply taking any action u that maximizes $Q^*(i, u)$ over $U(i)$.

The fundamental issue in reinforcement learning algorithm is how to efficiently find the optimal Q-factors. If the environment model (which includes the state transition probability function and the reward function) is available, traditional dynamic programming algorithms can be applied to obtain the optimal Q-factors [11]. In many practical problems, the environment model is usually unknown in advance. In the following subsection, a reinforcement learning algorithm called *Q-learning* which does not need the environment model is presented. It is used in all the reinforcement learning problems formulated in this thesis. It is a *model free* approach in that it does not try to learn the environment model. It simply tries to iteratively improve the estimation of the optimal Q-factors from the immediate reward received for each action taken. And based on these estimated optimal Q-factors, an optimal policy can be gradually derived.

4.2.1 Q-learning

Several algorithms have been proposed for the model-free reinforcement learning (RL) approach. First, let's assume that the environment is a fully observable Markov decision process that has finite state and action sets. Also, let's assume that the environment is stationary; that is, the state transition probabilities are not function of the stage number. One popular algorithm that is applicable to such an environment is the Q-learning algorithm by Watkins [114, 115]. It recursively estimates the optimal Q-factors ($Q^*(i, u)$) from experience obtained at every stage. The experiences are in the form of immediate reward sequence, $r(i_t, u_t, i_{t+1})$ ($t = 0, 1, 2, \ldots$). The aim is to obtain the optimal Q-factors for all state-action pairs based on which an optimal policy can be derived.

The Q-learning algorithm is analogous to value iteration method of dynamic programming [10]. However, it does not "sweep" through all the next possible states when updating the estimate of the optimal Q-factor $Q(i, u)$ of present state-action pair. For the discounted problem, the single-step sample update equation for $Q(i, u)$ is given as follows [114]:

$$Q_{t+1}(i_t, u_t) \leftarrow Q_t(i_t, u_t) + \alpha_t(i_t, u_t)[r(i_t, u_t, i_{t+1}) + \gamma \max_{u \in U(i_{t+1})} Q_t(i_{t+1}, u) - Q_t(i_t, u_t)] \tag{4.8}$$

where subscript indicates the stage number; $r(i_t, u_t, i_{t+1})$ denotes the immediate reward received due to the action u_t taken which causes the transition from state i_t to i_{t+1}; $\alpha \in [0, 1)$

```
INITIALIZE Q(i,u)
REPEAT (for each trial)
        INITIALIZE i
        REPEAT (for each step in the trial)
                Select action u under state i using policy (say ε-greedy) based on Q
                Take action u
                Detect new state i' and reward r
                IF i' not a failure state
                    Update Q-function:
                            Q(i,u) ← Q(i,u) + α[r + γ max_u' Q(i',u') − Q(i,u)]
                    i ← i'
                ELSE (r = r_f)
                    Update Q-function:
                            Q(i,u) ← Q(i,u) + α[r_f − Q(i,u)]
        UNTIL failure encountered or target achieved
UNTIL target achieved or number of trials exceed a preset limit
```

Figure 4-2: Q-learning algorithm based on lookup table[115, 102]

denotes the step-size parameter for the update; $\gamma \in [0,1)$ denotes the discount rate. Equation 4.8 updates $Q(i_t, u_t)$ based on the immediate reward $r(i_t, u_t, i_{t+1})$ and the maximum value of $Q(i_{t+1}, u)$ over all $u \in U(i_{t+1})$. This form of update is also called the bootstrapping [102].

For finite state and action problem, $Q(i,u)$ can be represented by a lookup table. The table is initialized with some values (e.g., zeros, random values, etc) before the beginning of the learning process. The Q-learning algorithm is summarized as in Figure 4-2. Several researchers [115, 111, 42] have proved the following convergence theorem for the Q-learning algorithm that uses lookup table representation for the Q-factors.

Theorem 4.2.1 (Convergence of Q-learning)
For a stationary Markov decision process with finite action and state spaces, and bounded rewards $r(i_t, u_t, i_{t+1})$. If the update equation (Equation 4.8) is used and $\alpha_t \in [0,1)$ satisfies the following criteria: $\sum_{t=1}^{\infty} \alpha_t(i,u) = \infty$ and $\sum_{t=1}^{\infty} [\alpha_t(i,u)]^2 < \infty$, $\forall (i,u)$; then $Q_t(i,u) \to Q^*(i,u)$ as $t \to \infty$ with probability 1, $\forall (i,u)$.

The theoretical convergence criteria is hard to be accomplished in practical implementation. Especially for large state and action spaces, the convergence rate for the Q-factors may be very slow because each state-action pair needs to be visited infinitely often in order to satisfy the convergence conditions. Hence, it is not practical for an application with large state and action spaces to adopt the Q-learning algorithm if an accurate value function is required to achieve given task. Q-learning works fine if the goal of the learning problem can be achieved without having a precise estimation of the optimal Q-factor for each state-action pair.

In the following subsubsection, two main implementation issues will be discussed. They are (1) the exploration versus exploitation; and (2) the credit assignment issues [102].

Exploration versus exploitation

In most reinforcement learning implementation, there is an issue concerning the trade-off between "exploration" and "exploitation". It is the balance between trusting that the information obtained so far is sufficient to make the right decision (exploitation) and trying to get more information so that better decisions can be made (exploration). For example, when a robot faces an unknown environment, it has to first explore the environment to acquire some knowledge about the environment. The experience acquired must also be used (exploited) for action selection to maximize the rewards.

If the agent always chooses the best ("greedy") action based on the current estimates of the Q-factors, it will result in poor exploration of the state-action space and the Q-factor estimates not converging to the optimal values. This is because the learning process involves the estimation of the optimal Q-factor of a state-action pair based on the estimates of the optimal Q-factor of other state-action pairs.

If the agent occasionally chooses a "suboptimal" action from the action set, it will acquire more information about those state-action pairs that are not visited at all or frequently. Exploration can also be achieved by randomly choosing a starting state in each trial. Once the Q-factors are sufficiently close to the optimal values, exploitation will result in an optimal action sequence.

One method that allows a structured exploration is called ϵ-greedy method [102]. In this method, the agent selects an action by maximizing the current set of $Q(i, u)$ over the action set with probability equal to $(1 - \epsilon)$ (where ϵ is usually a small number). Otherwise, the agent randomly (uniformly) selects an action from the action set $U(i)$.

The question is whether extensive exploration is necessary in a RL problem. If there is a strong need to achieve high quality estimate of the optimal Q-factors, it is important to have thorough exploration. However, for some applications, it is possible to derive a policy that is sufficiently good without having to explore the whole state-action space. Q-learning can be applied to this type of problems even if they have large state-action space.

For a control application in which most states have failure actions in the action set, it may not be feasible to explore randomly over the action set especially if many failures may damage the system. Even if the control system can withstand many failures, the exploration needs to be limited during an actual run. For the bipedal walking example, the biped may have to fall frequently if the exploration mode is not turned off or limited.

Another way to achieve exploration is by initializing Q-factors for all state-action pairs to optimistic values. At a given state, the learning agent selects an action that has the maximum Q-factor. If a failure is encountered or a penalty is given, the learning agent then downgrades the particular Q-factor of the state-action pair. In this way, when the same state is encountered in the future, the learning agent will select other actions. This type of exploration is called *passive* exploration since it is not achieved by adopting an exploratory policy.

Credit assignment

In a reinforcement learning problem, it is required to identify those actions that contribute to a failure (success) and properly discredit (credit) them. This is called the credit assignment or delayed reward problem. If an immediate reward function that correctly evaluates the present action taken can be obtained, it is not necessary to consider any delayed reward. If the reward function is not precise or if the action has an effect that lasts more than one stage, it is required to correctly assign credit to it based on the subsequent rewards received.

In the Q-learning algorithm, the discount factor γ determines the credit assignment structure. When γ is zero, only immediate reward (myopic) is considered. That is, any rewards generated by subsequent states will not have any influence on the return computed at the present state. Usually, if a reinforcement learning problem can be posed such that $\gamma = 0$ works fine, it is much simpler and can be solved more quickly.

When γ is large (close to one), the future rewards have significant contribution to the return computation for the present action. The agent is said to be farsighted since it looks far ahead of time when evaluating a state-action pair.

4.2.2 Q-learning Algorithm Using Function Approximator for Q-factors

This subsection describes the Q-learning algorithm that uses a function approximator to store the Q-factors. The function approximator (e.g., multi-layered perceptron, radial basis functions, CMAC etc.) is used as a more compact but approximate representation of $Q(i, u)$ [102]. The purposes of using a function approximator are as follows [93, 102, 10]:

- To reduce memory requirement; this is especially critical for high-dimensional and continuous state-action space.
- To enable generalization; generalization enables estimation of the Q-factors in those locations which are not explored.

There are two important requirements for an effective approximation [10]. First, the approximation architecture needs to be rich enough to provide acceptable approximation of the function. This usually requires practical experience or coarse information of the function (e.g., provided by theoretical analysis). Secondly, the algorithms for tuning the approximator's parameters need to be effective (e.g., low computational complexity).

Gaussian Radial Basis Function (GRBF) networks [15, 77, 76] and Cerebellar Model Articulation Controller (CMAC) [4] are common function approximators used in reinforcement learning algorithms. Both have nice local generalization property. It is desirable to have a function approximator with local, instead of global, generalization property in reinforcement learning problem because the former is less sensitive to the order of the training data. The order of the training data is usually determined by the dynamics of the overall system and thus it cannot be arbitrarily set. Although the GRBF Network has local generalization and is an universal approximator, its implementation is computationally costly unless further approximation is adopted. On the other hand, CMAC (see Section 4.3) has the advantage of having not only local generalization, but also being low in computation. The Q-learning algorithm using CMAC as the Q-factor function approximator is summarized as in Figure 4-3.

INITIALIZE weights of CMAC
REPEAT (for each trial):
    Initialize i
    REPEAT (for each step in the trial)
        Select action u under state i using policy (say ϵ-greedy) based on $\hat{Q}$
        Take action u,
        Detect new state i' and reward r
        IF i' not a failure state
            $\delta \leftarrow r + \gamma \max_{u'} \hat{Q}(i', u') - \hat{Q}(i, u)$
            Update weights of CMAC based on δ
            $i \leftarrow i'$;
        ELSE ($r = r_f$)
            $\delta \leftarrow r_f - Q(\hat{i}, u)$
            Update weights of CMAC based on δ
    UNTIL failure encountered or target achieved
UNTIL target achieved or number of trials exceed a preset limit

Figure 4-3: Q-learning algorithm using CMAC to represent Q-factors

Most reinforcement learning algorithms are useful only for those problems with finite state-action space unless function approximators are incorporated in the algorithms. However, there are very few theoretical analyses of the convergence conditions for such an approach. Boyan and Moore [13] included several counter examples of applying function approximators (mostly global type) to deterministic dynamic programming problems. They demonstrated by simulation that such an algorithm may be unstable. Nevertheless, Sutton [101] has successfully implemented the same test problems using the "SARSA" algorithm [2] with CMAC as the function approximator.

At present, the convergence conditions for most RL algorithms with function approximators are not known. However, for practical implementation, a function approximator is needed in most RL algorithms for the reasons stated earlier. In the thesis, the algorithm presented in Figure 4-3 is used for all the learning tasks.

4.2.3 Other Issues of Reinforcement Learning Implementation

Besides the issues concerning the credit assignment and the "exploration versus exploitation", there are other RL implementation issues. One other issue is the choice of a suitable reward function. The choice of the reward function is mainly an art and can greatly affect the performance of the learning. It requires the implementator to understand the problem at hand. For the example of the bipedal walking, a reward structure can be chosen such

[2] Almost similar to Q-learning but it is an on-policy algorithm.

that the agent is only penalized if the biped encounters a failure state and otherwise, it receives neutral reward (zero).

Another issue is the rate of learning, that is, how fast can the goal be achieved. One may achieve higher rate by using function approximators that possess local generalization property to store the value function. The generalization property is desirable for the agent to propagate any learned experience to the nearby states without experiencing all of them. Other approaches may also be used to increase the rate of learning. For example, the action space may be shrunken as much as possible based on prior knowledges. Learning speed can also be improved by detecting failure states earlier by using a tighter criteria for failure detection; or by increasing the successful space. For the biped walking, the proper usage of local control mechanisms to speed up the rate of learning will be demonstrated.

4.3 Cerebellar Model Articulation Controller (CMAC)

Cerebellar Model Articulation Control (CMAC) is a mathematical model proposed by Albus [3] for efficient computation in manipulator control. It is based on the neurophysiological theory of the cerebellum. In recent years, it has been adopted as a type of neural networks for supervised learning [68]. It receives lots of attention especially in the control field due to its local generalization and fast training speed. In this thesis, it is being used as a multivariable function approximator for the Q-factors in the Q-learning algorithm (see Section 4.2). There are many ways to implement a CMAC network. In this thesis, the original Albus's CMAC implementation is adopted. The corresponding programming code is adapted from that by Miller and Glanz [67].

The Albus's CMAC implementation (see Figure 4-4) mainly consists of an addressing scheme followed by a single-layered network. The addressing scheme maps the input vector to a proper subset (with C elements) of M weights of the single-layered network. By summing up the weights addressed, an output is obtained [3]. The addressing scheme is usually static throughout the training process. A supervised training algorithm is applied to adjust the weights of the single-layered network.

The addressing scheme consists of C layers of exhaustive partitioning of the input space. Each segment of a partitioning layer represents a receptive field. The partitioning layers are offset from one another. For a given input state, only one receptive field is activated for each partitioning layer. The total number of the receptive fields is thus equal to the total number (C) of the partitioning layers.

The shape of the receptive field can be anything. Rectangular type, which is adopted in the original Albus's CMAC implementation, is chosen in this thesis. This can be illustrated by considering a two-dimensional and continuous input space (x_1,x_2). And let's partition the state space uniformly to create grid-like receptive fields. Each of the receptive fields is linked to an index. This is like a lookup table if there is only one partitioning layer. If another layer (layer two) of receptive fields is overlapped and offset from the base layer as shown in Figure 4-5, a point (marked by a small circle) in the input space will yield two

[3]When the output is a vector, a single-layered network with the corresponding set of weights can be assigned to each of the output elements.

indices that correspond to the two activated receptive fields (rectangular boxes with bold border) from the layer one (base layer) and the layer two, respectively.

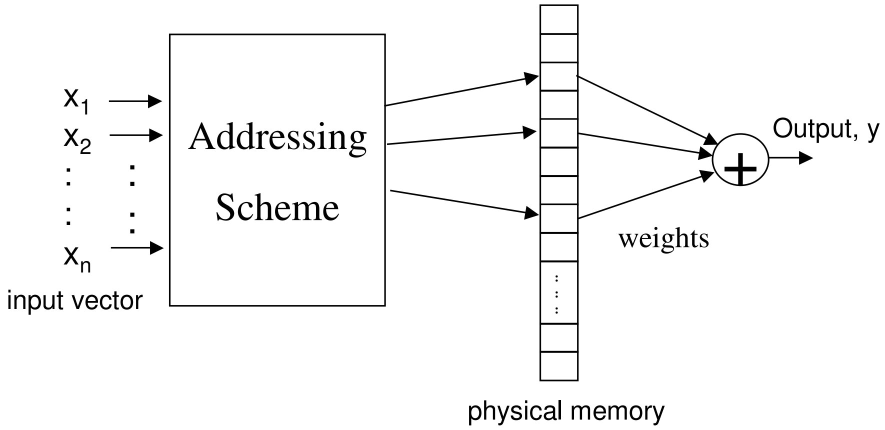


Figure 4-4: Schematic diagram of CMAC implementation

The widths (d_{x_1} and d_{x_2}) of the rectangular receptive fields are selected to match the desired generalization widths. The wider the receptive field, the greater is the generalization width. The number of partitioning layers corresponds to the desired accuracy in approximating a function. More layers means better approximation (finer discretization). However, having more layers also results in greater computational cost.

If the total number of receptive fields in each layer is large, *hashing* can be used to reduce the physical memory requirements. Hashing usually works by pseudo-randomly mapping multiple indices in a partitioning layer to a new index. This approach is plausible because in many cases, only a small fraction of the input space is important. Albus called this "input redundancy". The noise due to the overlaps of the hashing mapping will be small if the physical memory is sufficiently large and/or there is significant input redundancy.

Once the indices from the addressing scheme are generated, the output y (assume to be scalar) of CMAC can be obtained by summing C weights ($w^i : i = 1, \ldots, C$) linked to the indices:

$$y = \sum_{i=1}^{C} w^i \tag{4.9}$$

If the desired output y_d is given, these weights can be abjusted using the following equation which is similar to the LMS (Widrow-Hoff) algorithm [116]:

$$\delta w^i = \alpha \frac{(y_d - y)}{C}, i = 1, \ldots, C. \tag{4.10}$$

where δw^i is the adjustment to w^i and α is the training rate. That is, the adjustment is the same for all the weights. If α is large, the adjustment will also be large. However, large α

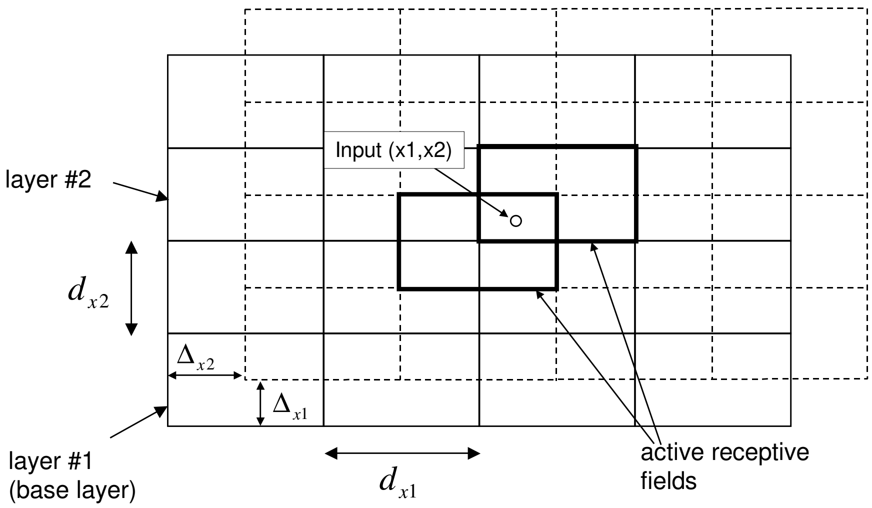


Figure 4-5: An addressing scheme for a two-dimensional input CMAC implementation

may lead to instability in the overall Q-learning algorithm that uses CMAC to approximate the Q-factors.

Note that some of the individual weights may diverge even though the output of CMAC is approaching the desired output. Miller and Glanz [67] have proposed adding extra term to Equation 4.10 to penalize large weight magnitude. This results in non-uniform adjustments for the weights for each training.

The key parameters for the Albus's CMAC implemention are the width of the receptive field along each of the state variables; the number of receptive fields layers and the learning rate α. They are adjusted by trial-and-error approach in the thesis.

Chapter 5

Sagittal Plane Control Algorithms

This chapter describes the control algorithm for the dynamic walking of the bipeds in the sagittal plane. The next section presents the analysis and heuristics of the walking motion in the sagittal plane. The analysis shows that the swing leg dynamics cannot be ignored. It also justifies the need for learning for the task. The learning algorithm is only applied to learn a parameter of the swing leg trajectory so that a walking goal can be reached. Three different algorithms and the detailed implementations will then be discussed in subsequent sections. This chapter also aims to establish the effectiveness of the stance ankle based velocity control, especially how it can be used to supplement the learning algorithm. The generality of the proposed algorithm is shown by applying one of the algorithms to two different bipeds that have different inertia and length parameters. The contrast between delayed and immediate reward implementations will also be explained.

5.1 Analysis and Heuristics of Sagittal Plane Walking Motion

From analysis done on human walking, Inman, Ralston and Todd [41] have listed two parameters, stride frequency and stride length, to be the key parameters for the forward speed control. In fact, the forward walking speed is simply equal to the product between the stride frequency and the stride length. This is true for any biped. However, this is a high level relationship and it does not lead to a solution for the biped walking control. A swing leg execution that is aimed to achieve specified stride frequency and length does not necessarily result in a stable gait. Furthermore, there are infinitely many different combinations of the stride frequency and length for a specified walking speed.

It is a well known fact that for constant-velocity walking or running, the acceleration during a stride must integrate to zero. Raibert [86] used this fact and the notion of symmetry to control the hopping speed of several running or hopping machines. The successful implementations were mainly due to the negligible legs' inertia and the symmetrical design. If the legs' inertia are not negligible or the symmetry plane is not so obvious, it will be difficult to implement this idea.

From the energy perspective, a single support phase of a dynamic walking consists of two different sub-phases. In the first sub-phase, the gravity is doing work against the motion, whereas in the second sub-phase, the gravity is pumping energy into the motion. So, if the swing foot lands at a location far ahead of the hip, the biped may not have sufficient kinetic energy to overcome the first sub-phase. If the swing foot lands at a location not sufficiently ahead of the hip, the biped will easily overcome the first sub-phase and may significantly accelerate forward during the second sub-phase. These phenomena can be illustrated using an inverted pendulum model.

In the past, most researchers tried to find an explicit solution for stable bipedal walking. So far, except for simplified models and/or systems, there is no explicit solution that is applicable to general bipedal walking. It is only known conceptually that the swing leg needs to be coordinated such that overall walking is stable.

There are several reasons why an explicit solution is not available. One obvious reason is that the sagittal plane motion is highly nonlinear. In addition, there are unpowered DOF between the stance foot and the ground [112]. A simple model that considers the unpowered DOF is the model used by Acrobot [73, 98] (see Section 5.1.2). The Acrobot model poses a great challenge to many researchers because it is not solvable by current methods in linear and nonlinear control theory [73]. Unlike in a fully actuated manipulator system, there is no direct solution that allows arbitrary trajectory tracking of all the state variables.

The following subsection describes two massless leg models for the sagittal plane bipedal walking. After this subsection, a double-pendulum model is used to illustrate the significant of the swing leg dynamics and the deficiency of massless leg models. The double-pendulum model is used to illustrate that the dynamics in the sagittal plane can be very complex even if a simplified model is used. This motivates the use of learning in the algorithm.

5.1.1 Massless Leg Models

A massless leg model is the simplest model for a biped in which the legs are assumed to have negligible inertia compared to the body. The swing leg dynamics are assumed to have little effect on the body motion. Thus, the swing leg motion can be planned independently without considering its effect on the body motion.

If the length of the stance leg and the body posture are assumed to be fixed during the single support phase and there is zero torque at the stance ankle, a simple point mass inverted pendulum model can be adopted as shown in Figure 5-1. The dynamic equations of this model can be derived by the Newtonian mechanics as follows:

$$\ddot{\theta} = \frac{g}{l}\sin\theta \tag{5.1}$$

where θ is the angle measured from the vertical, g is the gravitational constant, and l is the leg length.

This is a nonlinear differential equation which is difficult to solve analytically except using elliptic functions [100]. The phase portrait (Figure 5-2) of the inverted pendulum can be obtained by integrating Equation 5.1, and setting l and g to be $1m$ and $9.81m/s^2$, respectively. The phase portrait is symmetric about the abscissa and the ordinate axes. It has a saddle point at the origin ($\theta = 0$ and $\dot{\theta} = 0$) which corresponds to the point mass being at rest directly above the rotational joint. This is an unstable equilibrium point because any slight perturbation will cause the system to move away from it. The total energy (kinetic and potential) is *conserved*, and each continuous curve in the phase portrait corresponds to a constant energy level.

This model can be used to illustrate that if the swing foot lands at a location far ahead of the hip (say $\theta = -0.2rad$), the biped body may not have sufficient kinetic energy (say $\dot{\theta} = 0.3rad/s$) to proceed over the peak of the arc motion.

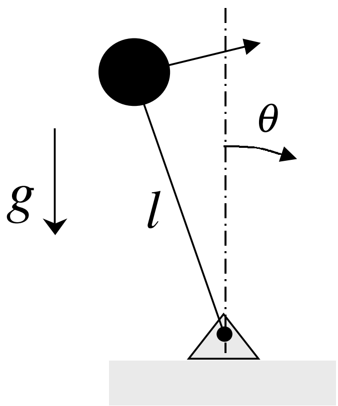


Figure 5-1: Massless leg model: Inverted pendulum. θ is the angle measured from the vertical, g is the gravitational constant, and l is the leg length.

Another massless leg model adopted by researchers [60, 46] is the *linear* inverted pendulum model. In this model, the walking height is assumed to follow a linear path by servoing the knee joint. For level ground walking, the height is kept constant. If the stance ankle is assumed to be limp, the point mass linear inverted pendulum model as shown in Figure 5-3 can be used. The dynamic equation of this model as follows:

$$\ddot{x} = \frac{g}{h} x \tag{5.2}$$

where x is the horizontal coordinate of the point mass from the vertical plane that passes through the ankle joint and h is the body height (see Figure 5-3).

Equation 5.2 is a linear homogeneous second order differential equation. Thus, an analytical solution is available as follows:

$$x(t) = C_1 e^{\omega t} + C_2 e^{-\omega t} \tag{5.3}$$
$$\dot{x}(t) = C_1 \omega e^{\omega t} - C_2 \omega e^{-\omega t} \tag{5.4}$$

where x is the body position; $\dot{x}$ is the body velocity; t is the time measured from the initial state; ω is given by $\sqrt{g/h}$; C_1 and C_2 are constants computed based on the initial state variables x_i and $\dot{x}_i$ to be:

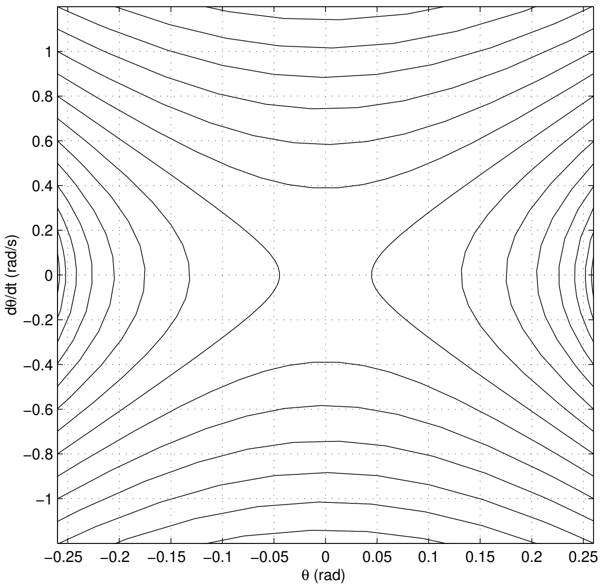


Figure 5-2: Phase diagram of the inverted pendulum model. Each contour corresponds to a specific energy level for the system.

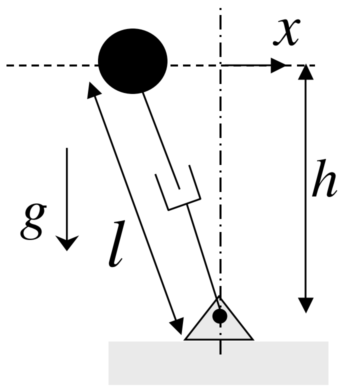


Figure 5-3: Massless leg model: Linear inverted pendulum. x is the horizontal coordinate of the point mass from the vertical plane that passes through the ankle joint.

$$C_1 = \frac{x_i + \dot{x}_i/\omega}{2} \tag{5.5}$$

$$C_2 = \frac{x_i - \dot{x}_i/\omega}{2} \tag{5.6}$$

The solutions can be converted to the following form by substituting the exponential terms with hyperbolic sine and cosine:

$$x(t) = x_i \cosh(\omega t) + \dot{x}_i/\omega \sinh(\omega t) \tag{5.7}$$

$$\dot{x}(t) = x_i \omega \sinh(\omega t) + \dot{x}_i \cosh(\omega t) \tag{5.8}$$

The linear inverted pendulum model is very attractive since an analytical solution is available. The model is usually used to generate a nominal trajectory for the body. The swing foot placement can then be planned according to the nominal body's trajectory. By fixing the time interval T_s for each step and assuming no ground impact force at the support exchange, Latham [60] used Equation 5.8 to compute the desired end position of the swing leg x_{swing} given the desired body velocity $\dot{x}_f$ at the end of the motion after the next step:

$$x_{swing} = x_i' = \frac{\dot{x}_f - \dot{x}_i' \cosh(\omega T_s)}{\omega \sinh(\omega T_s)} \tag{5.9}$$

where $\dot{x}_i'$ is the predicted velocity (based on Equation 5.8) at the next step.

In summary, the massless leg models can be used to illustrate key features and provide qualitative understanding of bipedal walking. In spite of these advantages, the following subsection will demonstrate that the swing leg dynamics cannot generally be ignored, and these models will not be applicable for dynamic bipedal walking control in the sagittal plane except in some limited cases.

5.1.2 Double-Pendulum Model

The main deficiency of the massless leg model is that it does not account for the swing leg dynamics which can be significant, especially during fast walking. It also does not apply to those robots that have significant leg inertia.

To illustrate the deficiency, a simple Acrobot model is adopted for the single support phase of the bipedal walking in the sagittal plane (Figure 5-4). The Acrobot is a two-link robot that is used to study the dynamics of a gymnast performing on a horizontal bar. It is actuated only at the joint between the two links. The joint that is attached to the ground is assumed to have zero torque.

This model is used to study underactuated mechanical systems [118, 73]. The Acrobot model poses a great challenge to control scientists because its motion control cannot be solved by present linear and nonlinear control theory. It will be used to demonstrate the deficiency of the massless leg model. It will also be used to justify the use of the learning approach instead of the pure model-based approach for the sagittal plane motion control of the biped.

The Acrobot model has two point masses M and m that represents the body and swing leg inertia, respectively. Both links have equal lengths ($l_1 = l_2 = l$). Let $\vec{\theta} = [\theta_1, \theta_2]^T$ be the generalized coordinates and $[\tau_1 = 0, \tau_2]$ be the corresponding generalized forces. The dynamic equations for the model can be derived using the Lagrangian formulation as follows:

$$h_{11}\ddot{\theta}_1 + h_{12}\ddot{\theta}_2 + c_1 + g_1 = 0 \tag{5.10}$$

$$h_{21}\ddot{\theta}_1 + h_{22}\ddot{\theta}_2 + c_2 + g_2 = \tau_2 \tag{5.11}$$

where

$$\begin{aligned}
h_{11} &= Ml^2 + m(2l^2 + 2l^2 \cos(\theta_2)) \\
h_{12} &= m(l^2 + l^2 \cos(\theta_2)) \\
h_{21} &= m(l^2 + l^2 \cos(\theta_2)) \\
h_{22} &= ml^2 \\
c_1 &= -ml^2 \sin(\theta_2)\dot{\theta}_2^{\,2} - 2ml^2 \sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 \\
c_2 &= ml^2 \sin(\theta_2)\dot{\theta}_1^{\,2} \\
g_1 &= -(m+M)lg \sin\theta_1 - mlg \sin(\theta_1 + \theta_2) \\
g_2 &= -mlg \sin(\theta_1 + \theta_2)
\end{aligned}$$

These equations are highly nonlinear even though it is a very simplified model for the single support phase of the biped. There is no analytical solution for these equations, and numerical methods are needed to solve them. Furthermore, there is an unactuated degree of freedom at the stance ankle. This prohibits the application of conventional nonlinear control theories [97].

This model will be first used to illustrate that the swing leg dynamics may affect the body dynamics and that the swing leg dynamics should not be ignored in the motion control. First, assume that the swing leg dynamics can be ignored and the swing leg trajectory is planned independently of the actual dynamics of the system. Let's consider the case in which the biped is travelling to the right. The stance leg starts from $\theta_1(0) = -15^o$ and the swing leg starts from $\theta_2(0) = -150^o$. That is, the angle between the front and trailing legs is 30^o. The swing leg is commanded to swing forward to $\theta_2(T_s) = -210^o$ based on a third degree polynomial function of time, where T_s is a predetermined swing time. The start and end angular velocities ($\dot{\theta}_2(0)$ and $\dot{\theta}_2(T_s)$) are equal to zero. That is, at the end of the swing, the angle between the legs will still be equal to 30^o.

A simple PD (Proportional and Derivative) control law is used to generate the torque τ_2 to track the desired θ_2. The proportional gain of the PD controller is set at $2000Nm$ and the derivative gain is set at $200Nms$. The body mass M is set at $10kg$ and the length l is set at $1.0m$. The swing time T_s is set at $0.7second$.

To illustrate the effect of the swing leg dynamics on the body motion, three phase

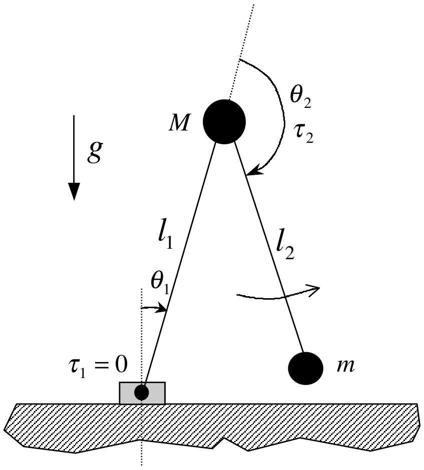


Figure 5-4: An Acrobot model is adopted for the single support phase of the bipedal walking in the sagittal plane. It has two point masses M and m that represents the body and swing leg inertia, respectively. It is actuated only at the joint between the two links. The joint that is attached to the ground is assumed to have zero torque.

portraits of θ_1 vs $\dot{\theta}_1$ (Figures 5-5(a), 5-5(b) and 5-5(c) correspond to the leg mass m of $1kg$, $2.5kg$ and $5kg$ respectively) are obtained numerically by taking the initial angular velocity of the stance leg to be from 0 to $2rad/s$ with an increment of $0.1rad/s$. From these phase portraits, it is observed that the symmetries about the abscissa and ordinate axes are lost especially for the case where $m = 2.5kg$ and $m = 5kg$. Also, from the trajectories of the stance leg angular position θ_1 for different leg mass m (0,1, 2.5 and $5kg$) (Figure 5-6), it is clear that the leg inertia cannot be ignored even for $m = 1kg$. Therefore, unless the swing leg task is carefully executed, the swing leg dynamics can severely affect the motion of the body and hence cannot be ignored in the bipedal walking control.

It is tempting to use the Acrobot model instead of the massless leg model for the bipedal walking control. However, since the dynamic equations are nonlinear and have underactuation, a control solution is not available unless linearization techniques are used. Most researchers chose to linearize the dynamic equations about the equilibrium points. Once this is done, many linear control theories can be applied. However, the linearized model is only applicable to motion control near the equilibrium points, which are usually motionless states. Thus, such a model cannot be applied to fast walking because all the nonlinear terms will become significant.

In the model-based approach, the parameter values need to be determined. This process is usually very tedious, especially if the model is complex. The unmodelled dynamics, for example, the actuator dynamics and the joint friction, may also cause the model to deviate from the actual system behavior. This can be illustrated with the Acrobot model as follows. Instead of studying the effect of the actuator dynamics, the effect of the control law dynamics is considered.

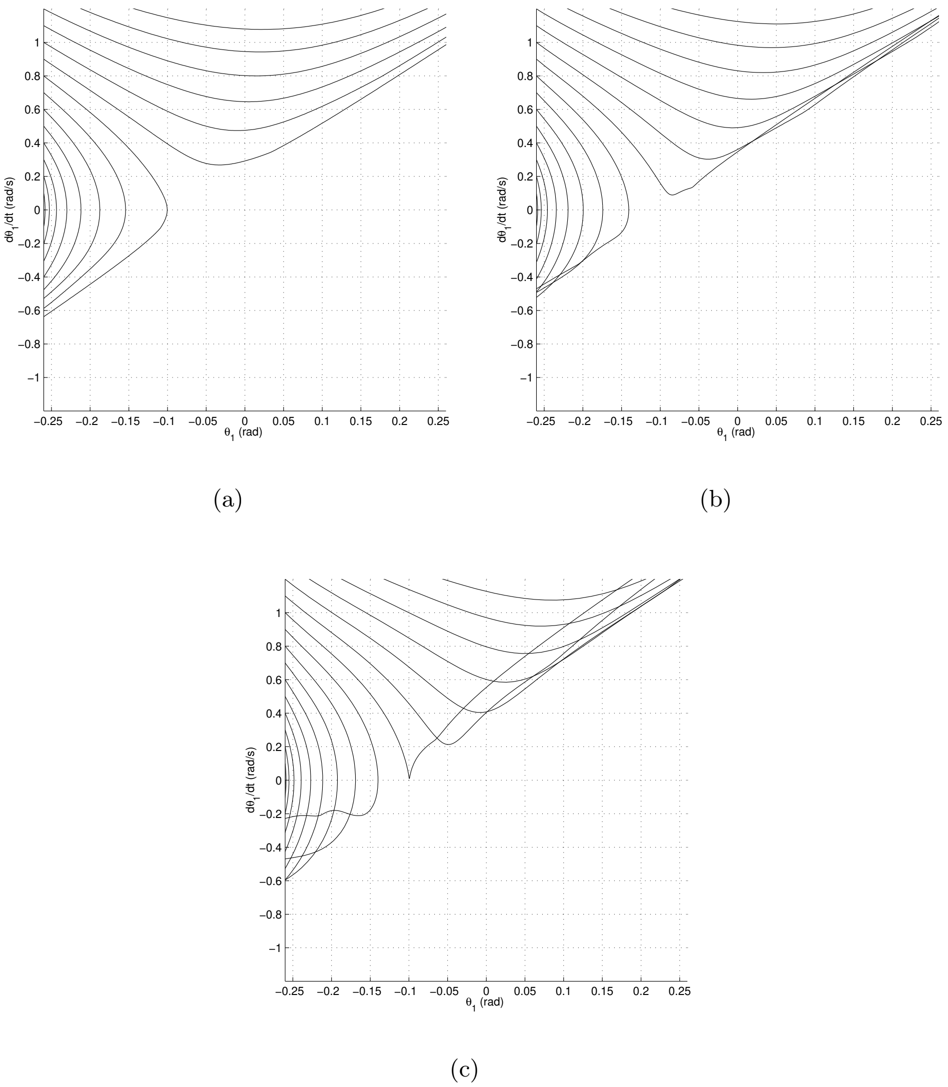


Figure 5-5: Phase portraits of the stance leg's angular position and velocity (based on the Acrobot model). The swing leg is commanded to travel from $\theta_2 = -150^o$ to -210^o in 0.7 second based on a third degree polynomial function of time (with zero start and end values for $\dot{\theta}_2$): (a) $m = 1kg$; (b) $m = 2.5kg$; and (c) $m = 5kg$.

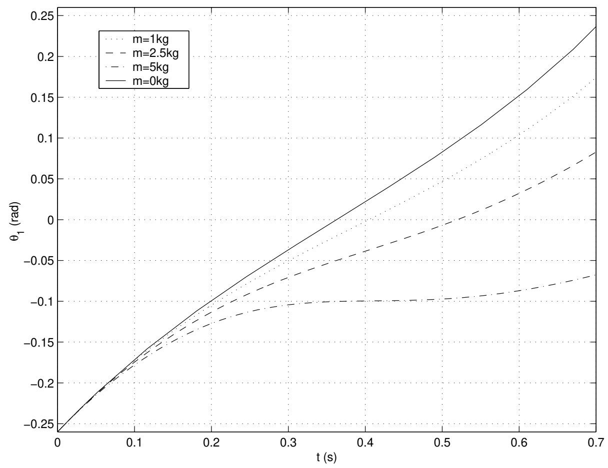


Figure 5-6: Trajectories of the stance leg angular position θ_1 based on the Acrobot model for different leg mass m (0,1, 2.5 and $5kg$). The initial values of θ_1 and $\dot{\theta}_1$ are $-15^o(0.26rad)$ and $57^o/s(1rad/s)$, respectively. The swing leg is commanded to travel from $\theta_2 = -150^o$ to -210^o in 0.7 second based on a third degree polynomial function of time (with zero start and end values for $\dot{\theta}_2$).

The body mass M and the leg mass m are set at $5kg$ and $2.5kg$, respectively. The initial values of θ_1 and $\dot{\theta}_1$ are $-15^o(0.26rad)$ and $57^o/s(1rad/s)$, respectively. As before, the swing leg is commanded to travel from $\theta_2 = -150^o$ to -210^o in 0.7 second.

The trajectory of the stance leg angular position θ_1 is obtained by the numerical simulation as shown in Figure 5-7. The solid curve corresponds to the trajectory of θ_1 where the PD controller has the proportional gain set at $2000Nm$ and the derivative gain set at $200Nms$. The dashed curve corresponds to the PD controller with the proportional gain set at 200Nm and the derivative gain set at 20Nms. The graph shows that the trajectories indeed deviate from one another.

From these numerical analyses, it can be deduced that the swing leg dynamics cannot be ignored for practical application to bipedal walking. Even if the swing leg dynamics are considered, other unmodelled dynamics may also cause the model to deviate from the actual behavior of the system. Furthermore, it is tedious to obtain the exact system parameters for the biped model.

Note that the Acrobat model does not consider the fact that the foot of the stance leg is not hinged on the ground. Thus, excessive torque between the two legs may not be achievable in the real biped. Furthermore, the biped needs to satisfy kinematics constraints, e.g., the swing foot must be above the ground or obstacles. These complex issues of bipedal walking motivate the use of the learning approach in the control algorithm. The next section describes the proposed control algorithm for the sagittal plane dynamic walking which is assisted by a learning algorithm.

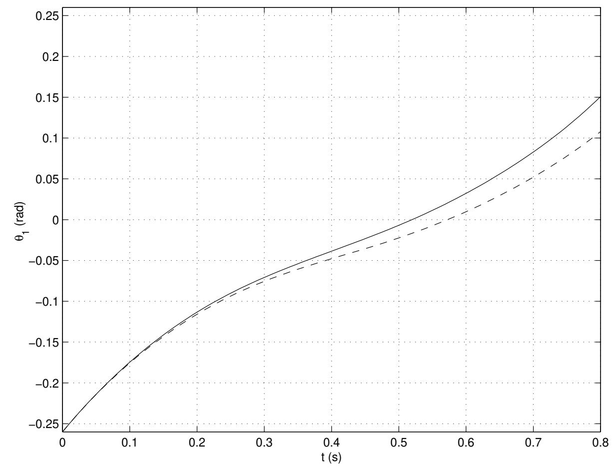


Figure 5-7: Effect of controller dynamics on the trajectory of the stance leg angular position θ_1. The swing leg is commanded to travel from $\theta_2 = -150^o$ to -210^o in 0.7 second as before. The solid curve corresponds to the PD controller with the proportional gain set at $2000Nm$ and the derivative gain set at $200Nms$. The dashed curve corresponds to the PD controller with the proportional gain set at 200Nm and the derivative gain set at 20Nms.

5.2 Algorithm

As mentioned earlier, the walking dynamics in the three orthogonal planes (sagittal, frontal and transverse) are assumed to be independent. The motion control in the sagittal, frontal and transverse plane can then be considered independently. This section presents a control algorithm for bipedal walking in the sagittal plane based on the divide-and-conquer approach introduced in Chapter 3. Here, the walking task can be divided into the following subtasks: 1) the height control of the body; 2) the pitch control of the body; and 3) the forward velocity control of the body. For the bipeds considered in this thesis, the joints whose axes are perpendicular to the sagittal plane are utilized for these subtasks.

Consider the “bird-like” and “bent-leg” walking posture and assume that the desired walking height and body pitch angle are constant. It is easy to achieve the first two subtasks using intuitive control approach like Virtual model Control (see Section 4.1) without the need of learning. For the horizontal speed control, the swing leg control approach is proposed instead of using the double support phase as in Pratt et al. [80]. Except for those bipeds that can be approximated by simplified models, there is no analytical solution for the desired swing leg behavior as a function of the desired walking speed. Thus, there is a need for learning, and a reinforcement learning algorithm (see Section 4.2) is chosen to learn the key parameter(s) for the swing leg control.

A finite state machine is used to control the state transitions of the system. The gait cycle of the biped is divided into four distinct states or phases: 1) right support phase; 2) right-to-left transition phase; 3) left support phase; and 4) left-to-right transition phase. Table 5.1 tabulates the descriptions of these states. A state diagram as shown in Figure 5-8 shows the transition events and flow directions among the states.

Table 5.1: Description of the states used in the bipedal walking control algorithm

State	*Description*
Right Support (RS)	The right foot is the stance leg and the left foot is the swing leg.
Right-to-Left Transition (RS_TO_LS)	This is the intermediate phase where the Right Support state transits to the Left Support state.
Left Support (LS)	The left foot is the stance leg and the right foot is the swing leg.
Left-to-Right Transition (LS_TO_RS)	This is the intermediate phase where the Left Support state transits to the Right Support state.

The following subsection describes the implementation of the control algorithm for the stance leg and the swing leg using the Virtual Model Control approach. The subsequent subsection describes the application of the reinforcement learning algorithm to learn the key swing leg parameters for the sagittal plane motion so that stable walking cycle can be achieved.

5.2.1 Virtual Model Control Implementation in Sagittal Plane

In the sagittal plane dynamic walking, the stance leg algorithm is mainly used to achieve the height control and body pitch control of the biped during walking. The swing leg algorithm is used to execute the swing leg strategy whose parameters are obtained by learning. The control algorithm for the stance leg and the swing leg can be considered separately as in the following subsubsections.

Stance Leg Algorithm

In the sagittal plane, the stance leg is used to control the body height and pitch. The height control is achieved by a virtual spring-damper attached vertically between the hip and the ground. It generates a virtual vertical force f_z at the hip. For the body pitch control, a virtual rotational spring-damper can be applied at the hip. This generates a virtual torque m_α about the hip. The virtual forces (f_z and m_α) can then be transformed into the desired torques for the stance knee joint and the stance hip pitch joint (τ_k and τ_{hp}, respectively) using a transformation matrix (Equation A.7 in Appendix A).

The parameters of the virtual components for the stance leg (sagittal plane) are summarized in Table 5.2. These parameters can easily be manually tuned, only requiring basic control intuition for a linear mass-spring-damper system. For example, for the vertical virtual spring-damper, it is desirable to have sufficiently high spring stiffness to minimize the positional error. The damping coefficient is then adjusted so as to achieve overdamped

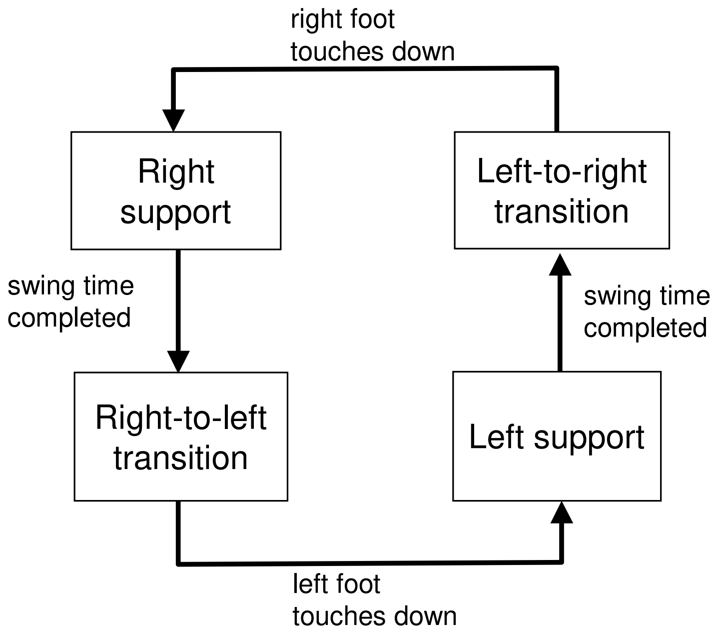


Figure 5-8: Finite state machine for the bipedal walking control algorithm.

response. A DC term is also added to counter the weight of the body so that a low spring constant [1] can be chosen for the virtual spring. The tuning process can be iterated if necessary. Such a tuning approach also applies to the rotational virtual spring-damper. Note that, although the virtual spring-dampers are linear components, the resulting dynamic system is nonlinear. Hence, it is not desirable to have oscillatory motion in the resulting system as nonlinear oscillatory motion can be non-intuitive.

[1] A low spring constant is desirable because a high spring constant may lead to instability in the physical implementation.

Table 5.2: Parameters of the virtual components for the stance leg (in the sagittal plane).

Parameter	*Notation*
1. Spring stiffness (in z direction)	k_z
2. Damping coefficient (in z direction)	b_z
3. Spring stiffness (in α direction)	k_α
4. Damping coefficient (in α direction)	b_α

Swing Leg Algorithm

There are many ways to implement the swing leg strategy. For example, one approach is to exploit the natural dynamics of the swing leg as in the passive walking approach [63, 82]. The desired hip pitch joint of the swing leg can be set to a value so that the servoing torque causes the swing leg to swing forward and the swing foot to naturally lift from the ground. The knee joint of the swing leg can be appropriately damped so that the swing leg lands onto the ground at the right moment. The advantage of this strategy is its low (mechanical) energy requirement. However, this strategy is not very versatile. For example, it is not suitable for fast walking control and for the case where step length control is needed (during difficult terrain walking).

This thesis mainly adopts a strategy for the swing leg that involves specifying a sequence of swing foot trajectory in the Cartesian space (see Section 5.2.2) as follows:

1. Lift the swing foot to a specific lift height, l_{lh}, from 0 to t_1 seconds. A third degree polynomial is used for this trajectory planning.

2. Swing the leg forward to a specific horizontal position once the foot is in the air. Again, a third degree polynomial is used for this trajectory planning.

3. After T_{swing} seconds, the swing foot starts to descend to the ground while maintaining its desired horizontal position.

Note that this is just one of the strategies that could be adopted for the swing leg. For example, a fifth degree polynomial may be chosen in order to specify no jerk condition [27]. Different strategies can also be adopted for the lifting and descending of the swing foot.

Given the sequence of the swing foot trajectories, two sets of linear spring-damper virtual components which are along the X-axis and the Z-axis of the inertia frame respectively can be adopted. Each of them is attached between the ankle of the swing leg and the trajectory path. The virtual forces in these virtual components are then transformed by a Jacobian equation (Equation A.4) to the respective joint torques of the swing leg.

Note that the swing leg strategy adopted has several parameters that need to be set. The gait stability of the dynamic walking depends on the set of parameter values chosen. In the next subsection, a reinforcement learning algorithm will be used to determine the values of selected parameters so that a walking goal can be achieved.

5.2.2 Reinforcement Learning to Learn Key Swing Leg Parameter

The key parameters for the swing leg strategy introduced in Section 5.2.1 are the lift height, swing time and the end position of the swing foot. The lift height is set to be constant throughout the walking to simplify the problem. That it, the only key parameters considered are the swing time and the end position of the swing foot. There are three main learning approaches. In the first approach, the swing time is fixed and the learning agent learns the end position of the swing foot. In the second approach, the end position of the swing leg is fixed and the learning agent learns the swing time. In the third approach, a learning agent learns the right combinations of these parameters.

In this thesis, the first two approaches are considered. The choice between the first and the second approaches depends on the terrain type. For example, if the actual foot placement or step length is not important (say in the case of the level ground walking), it is possible to simply set the swing time and learn the end position of the swing foot with reference to the body. However, if the actual foot placement is important (say in the case of the walking on stairs), it may be better to specify the desired step length and learn the swing time that results in stable gait.

This subsection presents the approach whereby the decision for the swing leg parameter can be posed as a discrete time delayed reward (or punishment) problem. The biped is supposed to select the right swing leg parameter at each step so that long term walking is possible. Momentarily after the the swing leg has landed, it detects a set of state variables and decides on an appropriate parameter value (from a finite set) for the swing leg strategy. This is a discrete-time dynamic system which can be represented by the following equation (assuming stationary):

$$i_{n+1} = f(i_n, u_n, w_n),\ n = 0, 1, \dots \tag{5.12}$$

where i_n is the state vector, u_n is the scalar control action (in this case the parameter value of the swing leg strategy), w_n is the random disturbance vector (in a countable space) and the subscript n denotes the stage number. The state vector i_n has continuous elements, whereas the control action u_n is an element of a finite set $U(i_n)$. The probability function of w_n is a function of i_n and u_n, but not a function of prior disturbances.

Starting from an initial state i_o, the objective is to generate a policy π (assumed stationary) so as to maximize the following expected cumulative discounted return:

$$V^{\pi}(i_o) = \lim_{N \to \infty} E[\sum_{n=0}^{N-1} \gamma^n r(i_n, \pi(i_n), w_n)] \tag{5.13}$$

subject to the constraint equation (5.12). r is the given scalar reward and γ is a discount factor.

It is assumed that the mapping function f is unknown. The biped needs to learn the appropriate parameter value of the swing leg strategy by trial-and-error until it can achieve long term walking without failure. The first step is to identify the state variables for the learning task. A reward function is also set up for the learning. After that, an appropriate reinforcement learning algorithm has to be chosen. The following subsubsections is devoted to these tasks.

State variables

The hip height and the body pitch are assumed to be constant during the walking and hence they can be excluded from the state variables set in the learning implementation. For constant desired walking speed (average), the following variables are identified to be the state variables for the learning implementation:

1. Velocity of the hip in the x-direction, $\dot{x}^+$;
2. x-coordinate of the previous swing ankle measured with reference to the hip, x^+_{ha} ;
3. Step length, l^+_{step}.

Superscript + indicates that the state variable is measured or computed at the beginning of a new single support phase.

The choice of the first two state variables are obvious. They represent the generalized coordinate for the body, assuming the height and the body pitch are constant. The third state variable (the step length, l^+_{step}) is required to cater for the dynamics of the swing leg (see Figure 5-9). If the swing leg dynamics were negligible, this state variable will be ignorable.

Note that this is just one particular set of state variables and there are many other possible sets that define the state of the biped for the sagittal plane motion. It may be required to include more state variables if the complexity of the walking behavior increases. For example, if the biped is walking on a slope, the gradient of the slope or the vertical distance between both feet may be included in the state variable set.

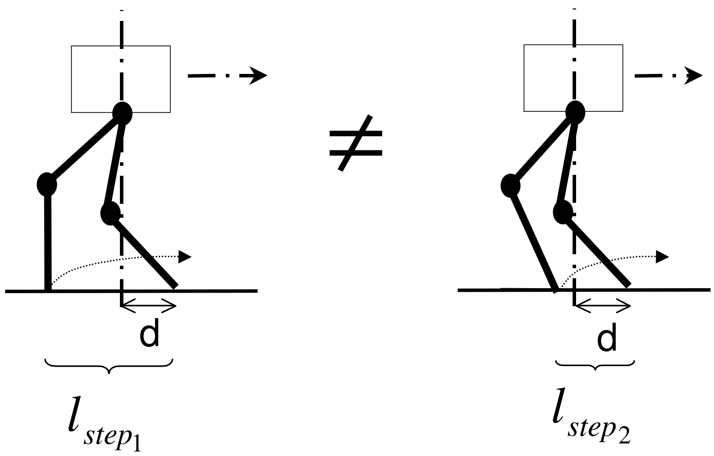


Figure 5-9: Two postures of the biped when the swing foot touches down. Though x^+_{ha} (x-coordinate of the previous swing ankle measured with reference to the hip when the swing leg touches down) is the same ($= d$) for both postures, the step lengths (l_{step_1} and l_{step_2}) are different. If the dynamics of the swing leg is not ignorable, the next swing leg dynamics for both postures will have different effects on the body motion.

Reward function and reinforcement learning algorithm

The biped aims to select a good value for the swing leg parameter for each consecutive step so that it achieves stable walking. A reward function that correctly defines this objective is

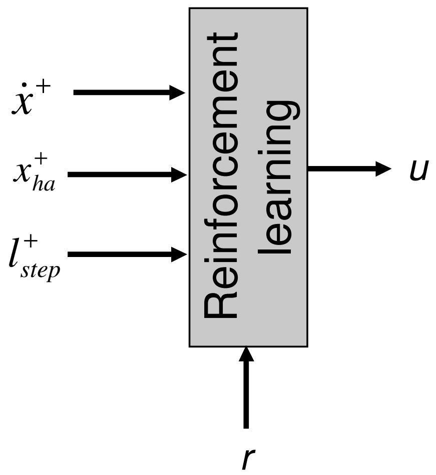


Figure 5-10: Schematic diagram of the RL implementation for the sagittal plane. $\dot{x}^+$ is the velocity of the hip in the x-direction; x_{ha}^+ is the x-coordinate of the swing ankle measured with reference to the hip; l_{step}^+ is the step length. Superscript + indicates that a state variable is measured or computed momentarily after the landing of the swing foot. r is the reward received and u is the selected parameter value of swing leg strategy.

critical in the reinforcement learning algorithm. If it is not correctly defined, the learning task may not converge at all. In formulating a reward function, there is usually a continuum of solutions. One needs to decide also how precise it should be.

Let's assume that there are minimum heuristics available for the implementation. A simple reward function can be formulated by defining a failure state to be one whose walking speed is above an upper bound V_u or below a lower bound V_l; or the hip height z is lower than a lower bound Z_l:

$$r = \begin{cases} 0 & \text{for } V_l \le \dot{x} \le V_u \text{ and } z \ge Z_l \\ R_f & \text{otherwise (failure).} \end{cases} \tag{5.14}$$

where R_f is a negative constant (punishment). That is, there is no immediate reward or punishment unless a failure state is encountered. This is a *delayed reward* problem and the agent has to learn to avoid the failure states and to avoid those states that inevitably lead to the failure states.

The schematic diagram of the reinforcement learning framework is shown in Figure 5-10. The Q-learning algorithm that uses CMAC network as the function approximator is adopted for the implementations.

The learning target is for the biped to achieve say 100 seconds of walking without encountering the failure states. Passive exploration (see Subsection 4.2.1) is achieved by initializing all Q-factors to an optimistic value, say zero. That is, all the weights of CMAC network are set to zero. Although the action sets encountered in this thesis can be continuous, they are discretized so that the search for the maximum Q-factor can be done without using an advanced search algorithm. The discretization resolution depends on the computational capability of the microprocessor and the nature of the problem. For example, in the case of the foot placement selection (see Section 5.3), the discretization resolution is chosen to be $0.01m$ that is fine enough so as it does not affect the walking behavior and require high computation.

The mechanism of the learning can be briefly summarized as follows. At the beginning of the first learning iteration [2], all the actions in the discrete action set will be equally good. In the event of a tie, the first action encountered in the search algorithm that has the maximum Q-factor will be adopted. Whenever a failure state is encountered, a new learning iteration is initialized. Before that, the Q-factor for the state-action pair that leads to the failure state will be downgraded. If all the actions at a give state lead to a failure state, the learning agent will avoid a state-action pair that leads to that state. As such, the learning agent gradually identifies those state-action pairs that lead to a bad outcome.

5.3 Effect of Local Speed Control on Learning Rate

In Section 3.3, the speed control mechanisms in the sagittal plane were divided into *local* and *global*. Although the local control mechanisms are not able to achieve the speed control by themselves, appropriate use of them may help to improve the performance of the control algorithm. In this section, the local speed control mechanism which is based on the stance ankle (Figure 5-11) will be demonstrated to significantly reduce the learning time.

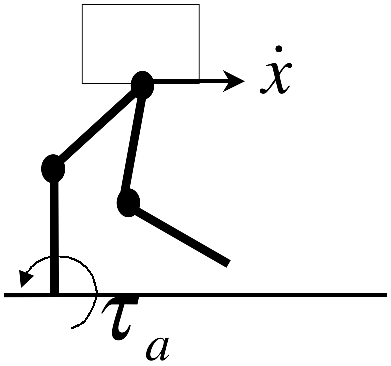


Figure 5-11: A local speed control mechanism based on stance ankle joint.

The control law applied to the stance ankle pitch joint to generate the required torque τ_a is as follows:

[2]Each learning iteration is from an initial state to either a failure state or any state in which the learning target is achieved.

$$\tau_a = B_a(\dot{x} - \dot{x}^d). \tag{5.15}$$

where B_a is a constant gain and superscript d indicates the 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. In the algorithm, a simple static analysis is used to generate the bounds:

$$\tau_{au} = K_{au} M g l_{at\_x} \tag{5.16}$$

$$\tau_{al} = -K_{al} M g l_{ah\_x} \tag{5.17}$$

where τ_{au} and τ_{al} are the upper and lower bound, respectively for τ_a; M is the total mass of the biped; g is the gravitational constant; l_{at_x} is the horizontal distance between the ankle and the toe; l_{ah_x} is the horizontal distance between the ankle and the heel; K_{au} and K_{al} are positive discount factors to compensate for any discrepancies. The bounds can be further optimized if better local control at the stance ankle is to be achieved. Even without optimization, the local control is demonstrated to be effective in improving the performance of the overall algorithm.

To test the effectiveness of the local control, a control algorithm was constructed and applied to the simulated Spring Flamingo. In the control algorithm, the stance leg and swing leg algorithms were implemented as described in Section 5.2.1. The swing time T_s was fixed at $0.3 second$. The Q-learning algorithm had to learn the desired end position of the swing foot. In this particular implementation, the desired position of the swing foot was measured horizontally from the hip. It is selected from a discrete action set U ($= \{0.01n|$ for $0 \leq 0.01n \leq 0.2$ and $n \in \mathbf{Z}\}$). The lower and upper bound of this discrete set are chosen to be 0.0 and 0.2, respectively. The lower bound of the action set is chosen mainly by observing the human counterparts. The human counterparts mostly swing the leg forward such that the swing ankle lands in front (horizontally) of the hip. As such, the lower bound is set to zero. The upper bound of the action set is chosen mainly based on geometric consideration. For example, it is not desirable for Spring Flamingo to set the desired foot placement (with respect to the hip) to be $0.5m$. Also, based on the physics of an inverted pendulum, if the swing foot lands far ahead of the hip, the body needs to have high velocity (kinetic energy) in order to proceed across the maximum potential energy point. Otherwise, the biped will fall backward. The values of V_u, V_l and Z_l used in the simple reward function (Equation 5.2.2) to determine the failure conditions are mainly chosen by using intuition. These parameter values (including the upper and lower bound for the action set) can be tuned by trial-and-error. The performance of the algorithm is not very sensitive to these values. For the simple reward function, a farsighted learning agent need to be used. Thus, the discount factor γ is set to a value close to one (say 0.9). The details of this implementation are summarized in Table 5.3.

The starting posture (standing posture) and initial walking speed ($0.4m/s$) were the same for every iteration during the learning process. The desired walking speed was $0.7m/s$. The simulation result is shown in Figure 5-12. The dashdot-line graph shows the simulation result corresponding to the case where local control was implemented. It achieves 100 seconds of walking at $16th$ iterations (each iteration corresponds to each attempt of the biped to walk until a failure condition is encountered). The solid-line graph corresponds to the result in which the ankle torque was always set to zero (limp joint). From comparison of both graphs, it is deduced that proper application of the ankle torque can speed up the learning rate for the walking.

The graphs in Figure 5-13 show the action (x^f_{ha}) sequences generated by the learning agent (with local control at the stance ankle) for the 1^{st}, 5^{th}, 10^{th}, 15^{th} and 16^{th} learning iterations. The corresponding forward velocity ($\dot{x}$) profiles are also included in the same figure. Except for the 16^{th} iteration in which the learning target is achieved, all other iterations prior to the 16^{th} iteration are terminated due to failure encounter. At the 16^{th} iteration, the first portions (before six seconds) of the graphs for both $\dot{x}$ and x^f_{ha} are rather chaotic. After that, the motion converges to a steady cycle.

Table 5.3: Reinforcement learning implementation for the sagittal plane: S_1

Description	Remark
Implementation code	S_1
Swing time, T_s	constant
Desired walking speed	constant
Learning output (action)	Horizontal end position of the swing foot with reference to the hip, x^f_{ha}
Learning target	$100s$ of walking
Key parameters:	
Reward function	Equation 5.2.2
Upper bound for the hip velocity V_u	$-0.1m/s$
Lower bound for the hip velocity V_l	$1.3m/s$
Lower bound for the hip height Z_l	0.5
R_f	-1
Discount factor γ	0.9 (farsighted)
Action set U	$\{0.01n \mid \text{for } 0 \leq 0.01n \leq 0.2 \text{ and } n \in \mathbf{Z}\}$
Policy	Greedy
CMAC parameters	
Width of the receptive field for $\dot{x}^+$	$0.205m/s$
Width of the receptive field for x^+_{ha}	$0.0384m$
Width of the receptive field for l^+_{step}	$0.1024m$
Width of the receptive field for x^f_{ha}	$0.0384m$
Number of the receptive fields layers C	128
Learning stepsize α	0.25

To verify that the local control at the stance ankle pitch joint does consistently result in a good learning rate, the same implementation that uses the local control is applied to

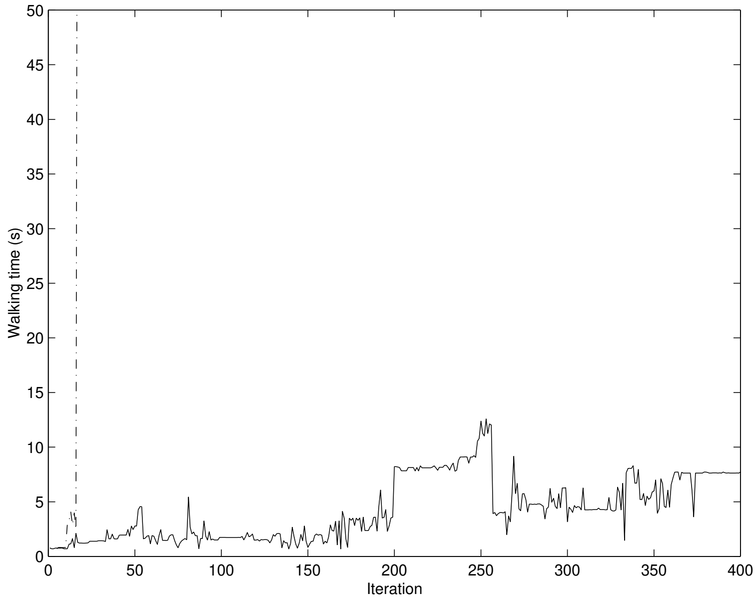


Figure 5-12: The graphs show that the local control based on the stance ankle increases the learning rate. The dashdot-line graph corresponds to the simulation result with the local control at the ankle. The solid-line graph is without the local control. In both cases, the starting posture (standing posture) and the initial speed ($0.4m/s$) are the same for each training iteration.

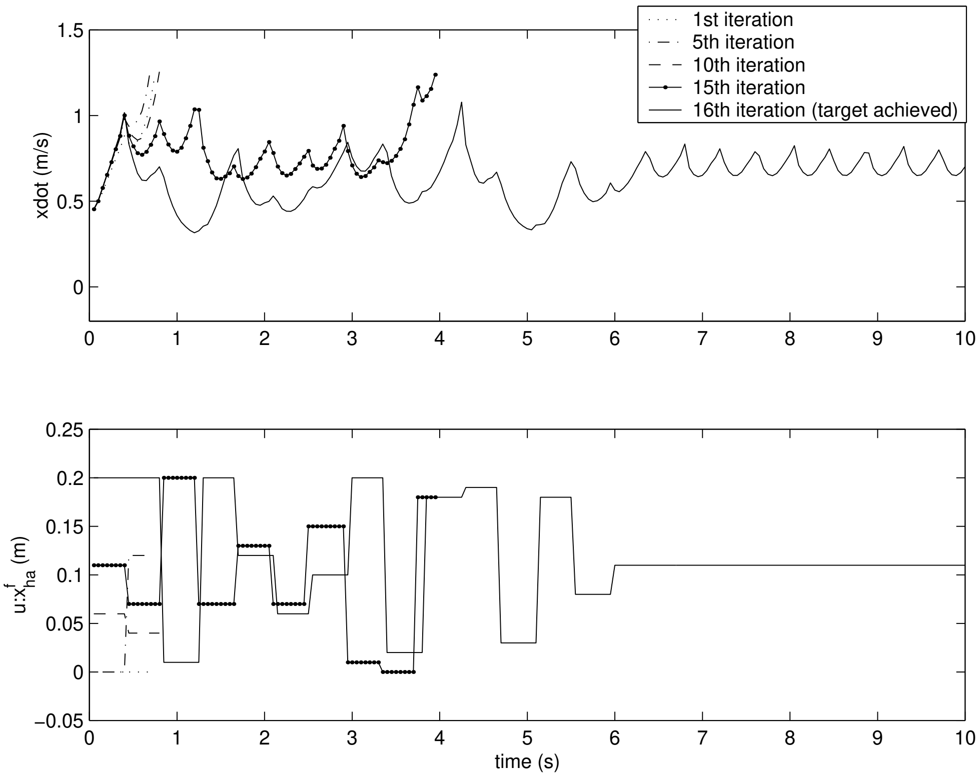


Figure 5-13: The graphs show the action (x^f_{ha}) sequences generated by the learning agent (with local control at the stance ankle) for the 1^{st}, 5^{th}, 10^{th}, 15^{th} and 16^{th} learning iterations. The corresponding forward velocity ($\dot{x}$) profiles are also included. The learning target is achieved at the 16^{th} iteration. The starting posture (standing posture) and the initial speed ($0.4m/s$) are the same for each training iteration.

the simulated biped for different swing times (0.2, 0.3, 0.4 and 0.5 second). The initial walking velocity is set to $0m/s$. All other parameters are set as before. Figure 5-14 shows the learning curves for each of the swing times. In the worst case, the biped can achieve 100 seconds of continuous walking within 90 iterations.

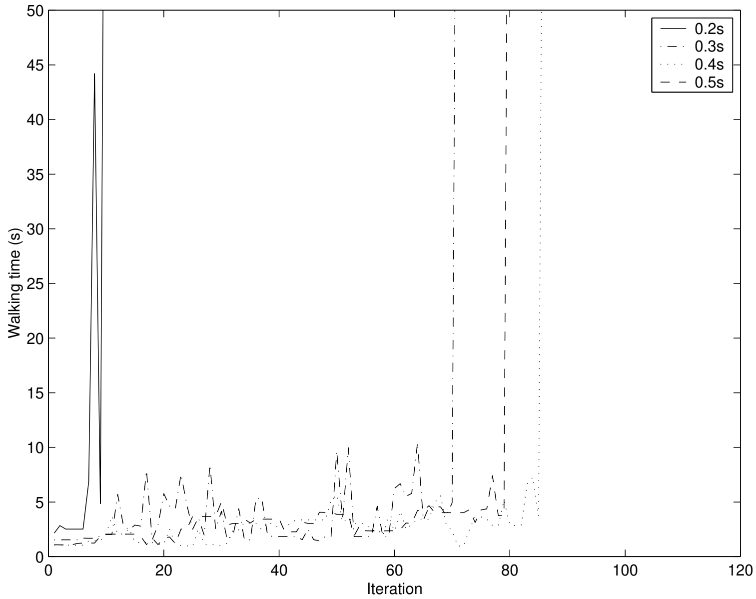


Figure 5-14: The learning curves of the simulated Spring Flamingo correspond to different swing times (0.2, 0.3, 0.4 and 0.5 second). The local control based on the stance ankle is used. The starting posture (standing posture) and the initial speed ($0m/s$) for each training iteration are the same for all the cases.

In summary, the proper usage of the local speed control mechanism based on the stance ankle helps to increase the learning rate. It is partly because it has increased the number of successful solutions. As such, the learning algorithm can find a solution much faster. The learning rate increase by the local mechanism may also be partly due to the fact that the mechanism possesses an "entrainment" property for the biped walking task. Thus, the learning agent need to simply generate a coarse foot placement position that can be "entrained" by the local control mechanism. The local control mechanism then fine-tunes the velocity control. Note that the biped is able to achieve the desired velocity quite well even though the reward function does not take it into account. All the subsequent implementations will use the local-control approach for the sagittal plane motion unless otherwise stated.

5.4 Generality of Proposed Algorithm

This section demonstrates the generality of the sagittal plane control algorithm in term of its applicability across bipeds of different mass and length parameters. In this implementation, the desired end position of the swing foot is specified based on the desired step length. The biped needs to learn the appropriate swing time, T_s, given the desired step length. For a given step length, Figure 5-15 illustrates that different swing times result in different postures, and hence the overall gait stability.

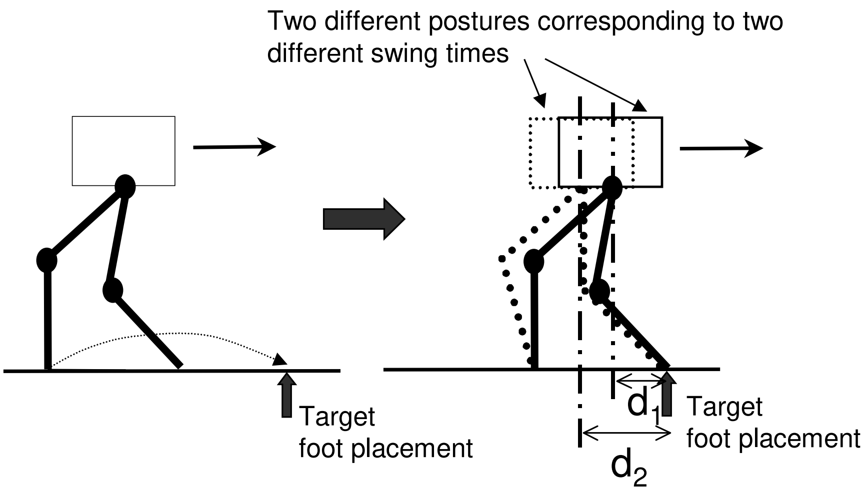


Figure 5-15: For a given foot placement location, different swing times yield different postures at the end of the step.

The control algorithm is applied to both simulated Spring Flamingo (SF) and M2 (planar). The desired step length is constant ($0.3m$). The starting posture (standing position) and the initial hip velocity ($0.4m/s$) are the same for each trial. The desired hip velocities are $0.7m/s$ for both bipeds. The desired heights are $0.74m$ for Spring Flamingo and $0.84m$ for M2. Again, the local speed control based on the stance ankle is adopted to assist in the velocity control. The results are shown in Figure 5-16. The learning speeds are comparable, and this demonstrates the generality of the proposed algorithm in term of its application to bipeds of different inertia and length parameters.

A stick diagram of the dynamic walking of Spring Flamingo is plotted in Figure 5-17. The simulation data for the Spring Flamingo implementation is shown in Figure 5-18. Those variables with "i" preceding them are the state variables for the reinforcement learning algorithm and that with "u" preceding it is the action variable. The top graph shows the states of the state machine (state 5 and state 6 correspond to the right support and the left support, respectively; state 7 corresponds to the right-to-left transition; and state 8 for the left-to-right transition). The horizontal dashed lines in the second and third graphs indicate the desired values.

From the second and third graphs, it is observed that the actual hip height and the forward velocity are well behaved, and they are close to the desired value. Especially for the forward velocity, although the reward function did not take into account the desired forward velocity, its average value converged quite well to the desired value. This outcome can be mainly attributed to the local control at the stance ankle (see Section 5.3).

Table 5.4: Reinforcement learning implementation for the sagittal plane: S_2

Description	Remark
Implementation code	S_2
Step length	constant
Desired walking speed	constant
Learning output (action)	Swing time T_s
Learning target	$100s$ of walking
Key parameters for RL algorithm:	
Reward function	Equation 5.2.2
V_u	$-0.1m/s$
V_l	$1.5m/s$
Z_l	$0.5m$ (SF)/$0.5m$ (M2)
R_f	-1
Discount factor γ	0.9 (farsighted)
Action set U	$\{0.02n \mid \text{for } 0.2 \leq 0.02n \leq 1 \text{ and } n \in \mathbf{Z}\}$
Policy	Greedy
CMAC	
Width of the receptive field for $\dot{x}^+$	$0.205m/s$
Width of the receptive field for x^+_{ha}	$0.0384m$
Width of the receptive field for l^+_{step}	$0.1024m$
Width of the receptive field for T_s	$0.0768s$
Number of the receptive fields layers C	128
Learning step-size α	0.5

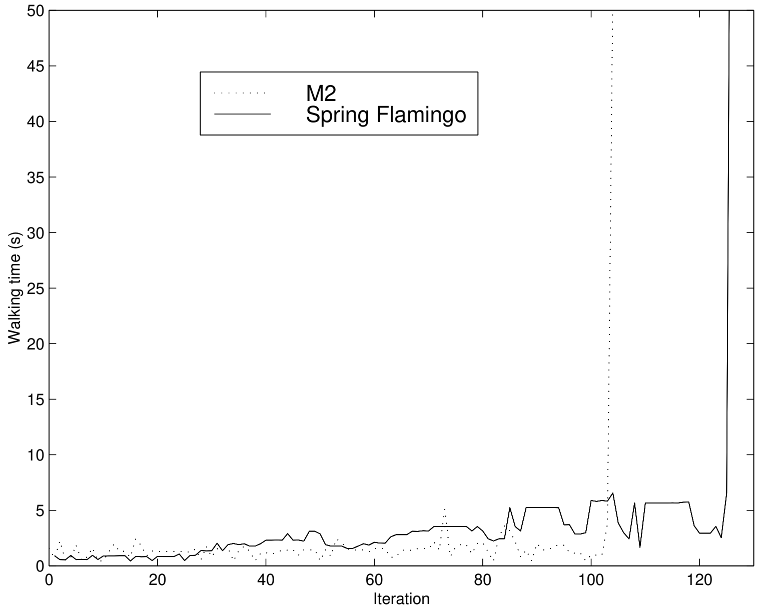


Figure 5-16: Learning curves for the simulated M2 (constrained to the sagittal plane) and Spring Flamingo (using implementation S_2).

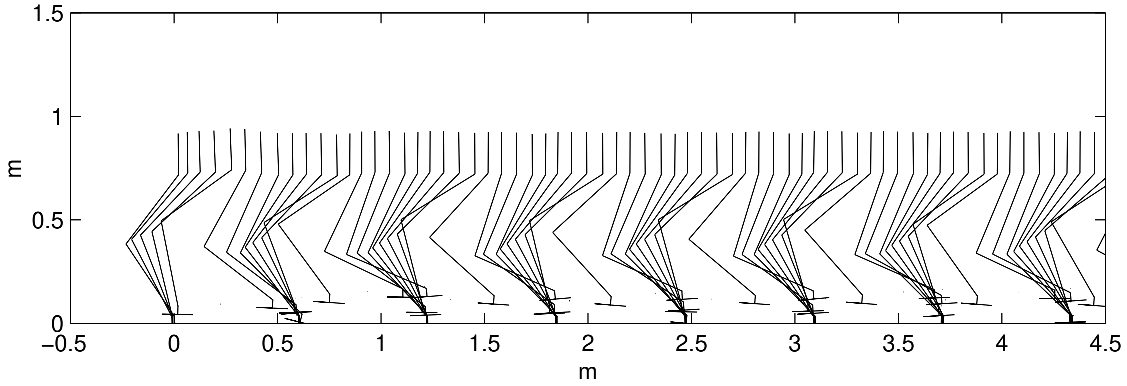


Figure 5-17: Stick diagram of the dynamic walking of the simulated Spring Flamingo after the learning target was achieved (using implementation S_2). Only the left leg is shown in the diagram, and the images are 0.1 second apart.

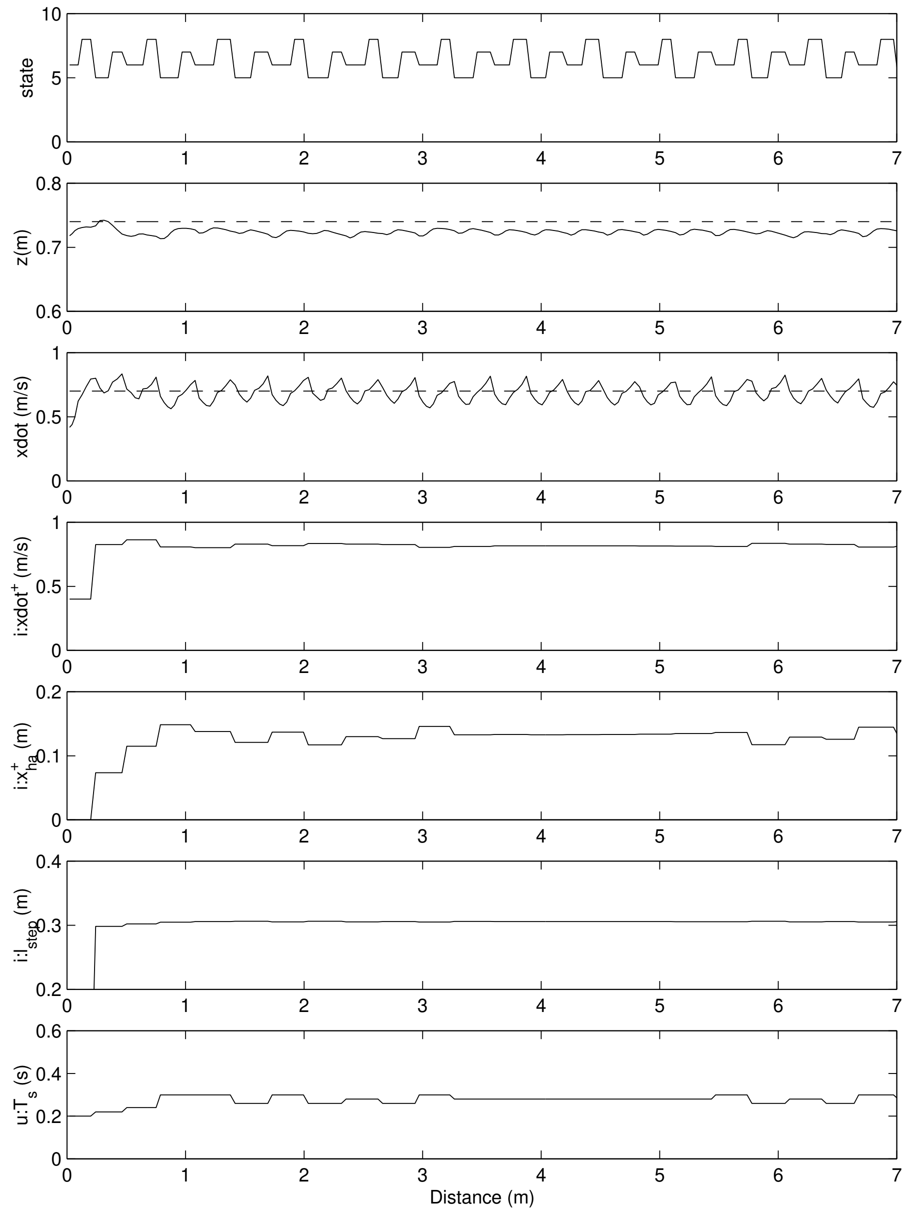


Figure 5-18: The simulation data for the dynamic walking of the simulated Spring Flamingo after the learning target was achieved (using implementation S_2).

5.5 Myopic Learning Agent

One shortcoming of the previous implementations (using delayed reward) for the sagittal plane is that it is good only for constant command variables like the desired velocity, step length etc. In general, it is desirable to change walking speed and to vary the step length while the biped is walking so as to avoid certain region (e.g. a pothole) on the ground.

To achieve these behaviors, the biped needs to evaluate its swing leg behavior based on an immediate reward rather than looking at the overall performance over several steps. In general, more heuristics or information is needed for the implementation. Sutton and Barto [102] called such a learning agent "myopic" (short-sighted) since it only tries to maximize the immediate reward.

The immediate reward need not be very precise. However, it must roughly represent how well the desired velocity is achieved after each swing leg control has been executed. A variable that represents the overall walking speed well is the hip velocity $\dot{x}'$ when the vertical projection of the biped's center of mass (VPCoM) passes through the stance ankle (Figure 5-19). VPCoM can be computed based on the kinematics information which includes the joint angles and the location of the center of mass (CoM) of each link. The mass of each link is also needed. The computation requirement is very low. Furthermore, great accuracy of the computation is not required.

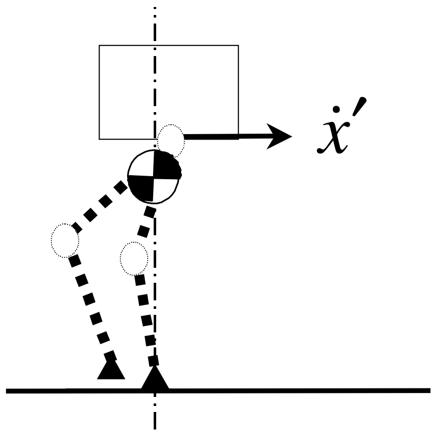


Figure 5-19: $\dot{x}'$ denotes the hip velocity when the vertical projection of the biped's center of mass passes through the stance ankle. This value is used to evaluate the action taken for the swing leg.

Except for those failure states, the reward function is computed based on the absolute error between $\dot{x}'$ and the desired velocity $\dot{x}_d$.

$$r = \begin{cases} -K_r|\dot{x}' - \dot{x}^d| & \text{for } V_l \leq \dot{x} \leq V_u \text{ and } z \geq Z_l \\ R_f & \text{otherwise (failure).} \end{cases} \tag{5.18}$$

where K_r is a positive constant.

Note that it is tempting to use the average velocity of the hip to compute the immediate reward. However, this is not a good performance indicator for walking. It is easy to see the reason. For example, when the velocity of the hip is monotonically increasing with an average value equal to the desired value during the single support phase, the hip velocity for the next cycle will be too high.

Table 5.5: Reinforcement learning implementation for the sagittal plane: S_3

Description	Remark
Implementation code	S_3
Step length	constant
Desired walking speed	constant
Learning output (action)	Swing time T_s
Learning target	$100s$ of walking
Key parameters for RL algorithm:	
Reward function	Equation 5.5
V_u	$-0.1m/s$
V_l	$1.3m/s$
Z_l	$0.7m$
R_f	-1
K_r	0.5
Discount factor γ	0 (myopic)
Action set U	$\{0.02n \mid \text{for } 0.2 \leq 0.02n \leq 1 \text{ and } n \in \mathbf{Z}\}$
Policy	Greedy
CMAC	
Width of the receptive field for $\dot{x}^+$	$0.16m/s$
Width of the receptive field for x_{ha}^+	$0.064m$
Width of the receptive field for l_{step}	$0.16m$
Width of the receptive field for T_s	$0.064s$
Number of the receptive fields layers, C	128
Learning step-size, α	0.5

This algorithm is implemented for the simulated M2 (constrained to the sagittal plane). The implementation details are summarized as in Table 5.5. The initial velocity for each iteration is set at $0.4m/s$. The desired velocity is $0.7m/s$ and the desired step length is $0.25m$. The biped needs to learn the swing time, T_s ($T_s \in U = \{0.02n \mid \text{for } 0.2 \leq 0.02n \leq 1 \text{ and } n \in \mathbf{Z}\}$) such that it can achieve 100 seconds of walking.

Figure 5-20 shows a learning curve for this implementation. The biped achieved the objective (100 seconds of walking) within 20 trials. The fast learning can be attributed to the immediate reward implementation. That is, if the learner's action is evaluated immediately rather than delayed, the learning process can be accelerated.

Figure 5-21 shows the key data during a dynamic walking of M2 after it has achieved the learning target. Those variables with "i" preceding them are the state variables for the reinforcement learning algorithm and that with "u" preceding it is the action variable. The top graph shows the states of the state machine (state 5 and state 6 correspond to right support and left support, respectively; state 7 corresponds to the right-to-left transition; state 8 for the left-to-right transition). The horizontal dashed lines in the second and third graphs indicate the desired values.

From the top second graph, it is observed that hip height is well-behaved (maintains a constant average value) though it falls short of the desired height ($0.84m$, dashed line). The error can be corrected by either increasing the DC offset for the vertical virtual component or applying Robust Adaptive Control in place of the virtual spring-damper for the height control [19].

From the third graph, the horizontal velocity $\dot{x}$ is also well-behaved though its average is slightly above the desired value ($0.7m/s$). From the second-to-last graph, it is observed that the step length, l_{step}, is greater than the desired value ($0.25m$). It is attributed to the overshooting of the swing leg control during the forward swinging phase. The precision of the swing leg trajectory tracking can be improved using Adaptive Control approaches used in manipulators [97].

From the last graph, it is observed that the swing time T_s for the first step is significantly greater than the rest. It is because the initial speed of the biped is $0.4m/s$. To increase it to $0.7m/s$, the biped needs to delay the touch-down of the swing leg until it has gained sufficient forward velocity.

In summary, this implementation produces good performance for the sagittal plane motion control. Such a myopic agent will be used for subsequent implementations in this thesis.

5.6 Summary

This chapter illustrates various ways of implementing the learning part of the proposed control architecture for the sagittal plane. Before the control algorithm is described, the need for learning in the dynamic walking implementation in the sagittal plane is first illustrated. After that, the Q-learning algorithm is applied to learn the parameter of the swing leg strategy in the sagittal plane. The Virtual Model Control approach is then used to generate the desired torques for all the joints in the same plane.

This chapter also illustrates the usage of the local speed control mechanism based on the stance ankle to significantly improve the learning rate for the implementation. The learning algorithm can be viewed as trying to identify the coarse failure zone for the action so that the biped would try not to select any action in this zone. The local control is used mainly to "entrain" the biped to achieve the desired walking speed. Thus, the local control can be viewed as trying to fine-tune the behavior. Note that if the local control mechanism is not implemented in the right way, it may worsen the overall performance of the algorithm. This will be illustrated in the next chapter.

Although only the stance ankle pitch joint is used to modulate the walking speed, other local control mechanisms can be added to further enhance the walking stability and robustness in the sagittal plane. For example, if the walking speed of the biped is too slow, a delay may be added to the support-exchange.

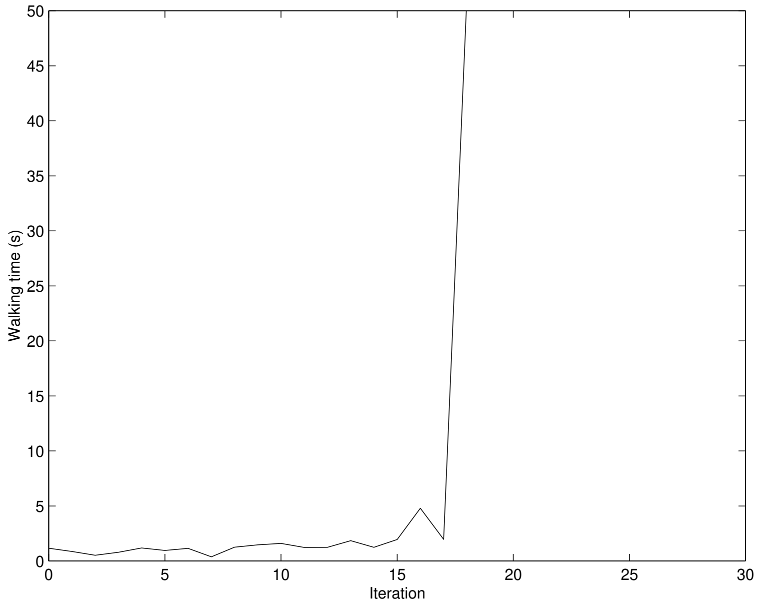


Figure 5-20: A learning curve for the simulated M2 (constrained to the sagittal plane) when implementation S_3 was used.

Next, the generality of the algorithm is demonstrated. The algorithm can be applied without major retunings across different bipeds that have the same structure but different inertia and length parameters. Each biped simply goes through its learning process to achieve the walking motion in the sagittal plane.

The theory of the Q-learning algorithm requires that all the state-action pairs (discrete case) be visited infinitely many times so that their Q-factors can converge to the optimal Q-factors. This requires proper exploration of the state-action space. In this chapter, it is demonstrated that for the Q-learning implementation in the walking algorithm, the learning goal can be achieved even without randomly exploring the state-action space (ϵ is set to zero for the ϵ-greedy policy). That is, for the bipedal walking task formulated in this chapter, it is not necessary to obtain a good approximation of the optimal Q-factors.

In the Q-learning implementation, CMAC was used as a function approximator for the Q-factors. It speeds up the learning rate of the algorithm. Besides this, there are other ways to speed up the learning process. One approach is to reduce the search space. The search space for the action can be reduced by changing the boundaries after knowledge has been gained for the task. The boundaries may also be functions of the state variables instead of being the same for all the states. The functions need not be very precise. Of course, the more precise the functions are, the tighter the boundaries can be set. Once the search space is reduced, the learning process should further be shortened. On the other hand, the successful space for a given task can be increased. The usage of the local control mechanism is equivalent to increasing the successful space.

The last implementation in this chapter adopted a myopic agent for the learning al-

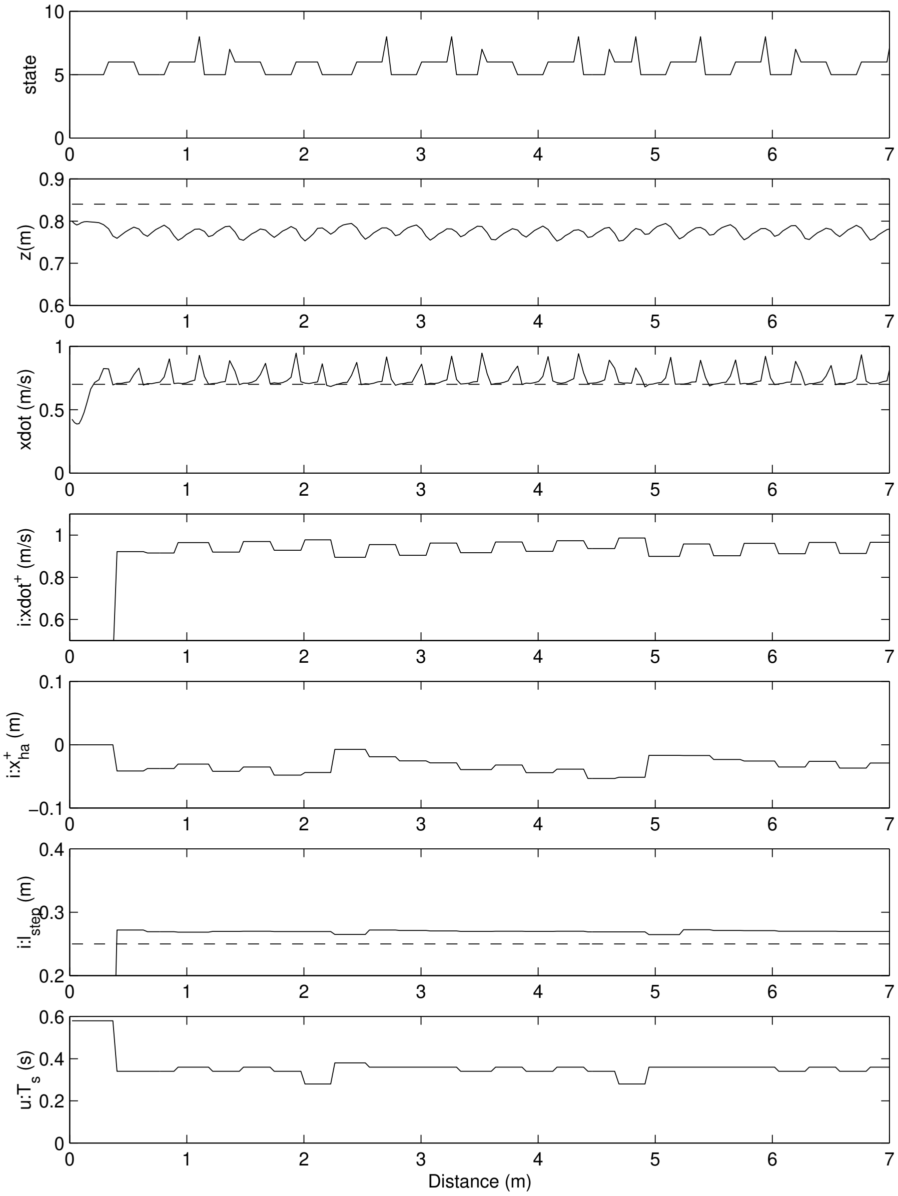


Figure 5-21: The simulation data for the dynamic walking of the simulated M2 (constrained to the sagittal plane) after the learning target was achieved (using implementation S_3).

gorithm. In the implementation, a reward was assigned to each decision and the rewards associated with any subsequent decisions were ignored. This implementation is used if the biped needs to change its foot placement location or its walking speed on the fly. The trade-off is that it requires extra information, in this case, the position of the center of mass.

Table 5.6 gives a summary of the implementations in this chapter. They have been successfully implemented for the planar dynamic walking of the bipeds without using any dynamic model. The next chapter will describe the frontal plane algorithm and how it can be combined with the sagittal plane algorithm to yield an algorithm for the three-dimensional bipedal walking control.

Table 5.6: Summary of the sagittal plane implementations

Code	End Position of swing leg		Swing time	Return computation
	From hip	Step length		
S_1	Learn	-	Fixed	Farsighted
S_2	-	Fixed	Learn	Farsighted
S_3	-	Fixed	Learn	Myopic

Chapter 6

Three-dimensional walking

This chapter describes the control algorithms for the frontal and transverse planes. It is assumed that the motion in the three orthogonal planes can be independently considered. First, the heuristics of the frontal plane motion will be presented. This is followed by descriptions of the algorithms for the frontal and transverse planes motion control. In the subsequent sections, different frontal plane implementations are studied. The use of the local control mechanism, based on stance ankle roll joint, for the frontal plane will also be explored. The algorithms for the frontal, transverse and sagittal planes are then combined and applied to the simulated M2 to achieve dynamic walking.

6.1 Heuristics of Frontal Plane Motion

In the frontal plane, the biped uses lateral foot placement to maintain its lateral balance. During steady forward walking, the body's lateral velocity varies periodically like a sinusoidal waveform [41]. The average lateral velocity has to be zero if the biped follows a straight line walking path. At each step, the lateral velocity of the body is expected to change sign exactly once.

If the swing leg dynamics in the frontal plane are neglected and the height of the biped is maintained at a constant value, the frontal plane motion can be coarsely modelled using the linear inverted pendulum model described in Subsection 5.1.1. Since the linear inverted pendulum model is a coarse model, it is not utilized to plan the swing leg motion. Instead, it is utilized only in the computation of the failure boundaries and for the local control implementation (see Section 6.3).

The governing dynamic equation for the linear inverted pendulum model is given as:

$$\ddot{x} = \frac{g}{h}x \tag{6.1}$$

where x is the horizontal coordinate of the body from the vertical plane that passes through the ankle joint, g is the gravitational constant and h is the body height (see Figure 5-3).

Equation 6.1 can be rewritten as follows:

$$\dot{x}\frac{d\dot{x}}{dx} = \frac{g}{h}x \tag{6.2}$$

The above equation can be integrated to give the relationship between $\dot{x}$ and x:

$$\frac{\dot{x}^2}{2} = \frac{gx^2}{2h} + C \tag{6.3}$$

where C is the integration constant which Kajita, Tani and Kobayashi [51] call the *orbital energy*. Given the velocity $\dot{x}$ and the position x of the system at any instant, C can be computed, and Equation 6.3 defines the relationship between $\dot{x}$ and x for all other time before the next support exchange event.

When C is greater than zero, the mass approaching the vertical plane that passes through the pivoting point will be able to travel across the plane. When C is less than zero, the mass will not be able to travel across the vertical plane and revert its direction of travel at some instance of time. In the frontal plane implementation, C is desired to be less than zero because it is desirable for the body's lateral velocity to change sign within the step.

Since this model is a coarse model for the frontal plane, it is only used for the failure detection in the learning algorithm. This will be explained in detail in Subsection 6.2.2. The model is also used in the frontal plane local control based on the stance ankle roll joint (see Section 6.3).

6.2 Frontal and Transverse Planes Algorithms

This section describes the control algorithms for the frontal and transverse planes. They are designed for the biped walking along a straight path. The Virtual Model Control approach is used to generate the desired torque for relevant joints whereas the reinforcement learning method is used to learn the key parameters, if necessary, of the algorithms.

The algorithms in this section will then be combined with the algorithm developed in the previous chapter to yield a three-dimensional walking algorithm. Again, the objective of the learning is to achieve 100 seconds of walking without encountering the failure states.

6.2.1 Virtual Model Control Implementations in Frontal and Transverse Planes

This subsection describes the algorithms for the frontal and transverse planes based on the Virtual Model Control approach. The algorithms are also modulated by the finite state machine described in Subsection 5.2.1. Each of the algorithms can be decomposed into the stance leg and swing leg sub-algorithms as explained in the following subsubsections.

Stance Leg

In the frontal plane, the desired body roll angle is set to be zero (upright posture) and it can be regulated by installing a rotational virtual spring-damper at the stance hip roll joint. A DC term can be added to account for the moment due to the body and the swing leg weights so that high stiffness is not required for the joint. Similar to the sagittal plane implementations, the stance ankle roll joint is used to provide a local velocity control for the frontal plane motion (see Section 6.3).

In the transverse plane, the main concern is the body rotation in the yaw direction. In all the implementations, the desired yaw angle of the body is set to zero. That is, the biped is controlled to face forward while it is walking. This can be achieved by adding a rotational virtual spring-damper at the stance hip yaw joint. The resulting effect is like a PD controller.

Swing Leg

For the swing-leg control in the frontal plane, the robot needs to know the desired trajectory of the swing-hip roll angle and the swing-ankle roll angle. There are several ways to obtain the trajectory of the desired swing-hip roll angle. They will be discussed later in this chapter. The desired ankle roll angle is obtained by geometric consideration. For example, the swing foot is maintained to be parallel to the ground when the foot is in the air.

Once these angles are given, rotational virtual spring-dampers are attached to the swing hip and ankle roll joints to track them. The next task is to simply choose a working set of the parameter values for the virtual components.

In the transverse plane, a rotational virtual spring-damper at the swing-hip yaw joint is used to regulate the swing leg yaw motion so that the swing foot is always pointing in the forward direction. Again, the key task is to choose a working set of the parameter values for the virtual components.

6.2.2 Reinforcement Learning to Learn Key Swing Leg Parameter for Lateral Balance

The lateral stability of bipedal walking is determined mainly by the swing leg behavior in the frontal plane. This subsection describes two strategies for the swing leg in the plane. Based on these strategies, the trajectory for the swing-hip roll joint can be generated.

In the first strategy, the desired end position ϕ^f (with respect to the vertical plane) of the swing leg in the frontal plane is generated before the beginning of the next step using the reinforcement learning method. The trajectory of the roll angle ϕ of the swing leg is then planned using a third degree polynomial. The swing time is assumed to be specified by the sagittal plane implementation.

The second strategy is based on the notion of symmetry. The desired swing leg roll angle ϕ^d is set to be equal to the actual stance leg roll angle θ. The reinforcement learning method is used to learn the offset Δ to the nominal desired swing leg roll angle provided by the symmetry approach (see Section 6.4).

Q-learning using CMAC as the function approximator for the Q-factors is adopted as the reinforcement learning method for both implementations. The learning aims to prevent the biped from encountering failure states in the frontal plane. The following subsubsections describe the state variables and reward function used in these strategies.

State variables

The frontal plane motion control is assumed to be weakly coupled to the sagittal and transverse planes motions. Thus, the state variables for the sagittal and transverse planes are not considered in the frontal plane learning implementation. In this thesis, the following state variables are chosen (see also Figure 6-1):

1. The velocity of the body in the y-direction, $\dot{y}^+$;
2. The roll angle of the new swing leg (with respect to the vertical), ϕ^+;
3. The roll angle of the new stance leg (with respect to the vertical), θ^+.

The superscript + indicates that the variable is detected at the end of the support exchange (just before the beginning of a new step).

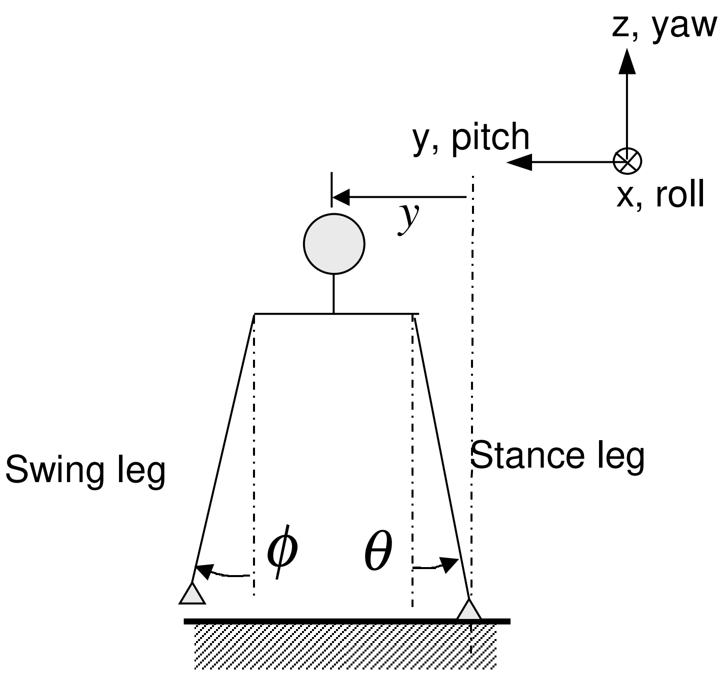


Figure 6-1: State variables for the learning implementation in the frontal plane motion are the values of y, ϕ and θ at the end of the support exchange (just before a new step begins).

Reward function and return computation

One way to implement the reinforcement learning algorithm is to adopt a reward function that only issues a punishment value when a failure state is encountered. In this case, a "farsighted" approach for the return computation has to be adopted. That is, the discount rate γ_2 for the return computation is set to a value close to one (say 0.9). In the frontal plane implementation, the following reward function is used:

$$r_2 = \begin{cases} 0 & \text{for } \Theta_l \leq \theta \leq \Theta_u \text{ and } C_l \leq C \leq C_u \\ R_{f2} & \text{otherwise (failure).} \end{cases} \tag{6.4}$$

where θ is the stance leg angle (Figure 6-1) bounded by Θ_l and Θ_u; C is the orbital energy bounded by C_l and C_u; and R_{f2} is a negative constant (punishment). Note that the failure condition based on θ is checked throughout each step. The failure condition based on C is checked only at the end of the support exchange.

6.3 Local Control in Frontal Plane

This section describes the use of the stance ankle's roll joint as a local control mechanism to assist the learning algorithm in the frontal plane. Two local control law candidates will be compared. The first control law candidate for the stance ankle roll joint (right support phase) is given as follows:

$$\tau_{ar} = -B_{ar}\dot{y} \tag{6.5}$$

where τ_{ar} is the desired torque of the stance ankle roll joint (acting on the foot by the shank), B_{ar} is a constant gain and $\dot{y}$ is the lateral velocity of the body. This control law is chosen because the biped average lateral velocity is desired to be zero. It tries to reduce the lateral velocity to zero at all times.

The other local control law candidate is built upon a coarse model for the frontal plane. The linear inverted pendulum model described earlier (see Subsection 5.1.1) is selected to be the coarse model. It has the following explicit solutions given by Equation 5.3 and 5.4:

$$\begin{aligned} y(t) &= C_1 e^{\omega t} + C_2 e^{-\omega t} \\ \dot{y}(t) &= C_1 \omega e^{\omega t} - C_2 \omega e^{-\omega t} \end{aligned}$$

where y is the horizontal position of the midpoint between the hips measured from the stance ankle, $\dot{y}$ is the time derivative of y, ω is equal to $\sqrt{g/h}$, C_1 and C_2 are constants that can be computed given the value of y and $\dot{y}$ at a certain time.

Based on the linear inverted pendulum model, the desired frontal plane motion during the right support phase can be represented as in Figure 6-2. Let's assume that the height h is constant. The black circle represents the point mass of the linear inverted pendulum model. The figure illustrates that the body changes the direction of motion (lateral) somewhere during the right support phase. This is possible only if the orbital energy C is less than zero (see Section 6.1). For a given swing time T_s, it is desirable by symmetry that the lateral velocity $\dot{y}$ of the body be zero at half time $T_s/2$. Furthermore, for regular walking, it is desirable that $y(T_s/2)$ be the same for every step. Once a value for $y(T_s/2)$ is chosen, the values of C_1 and C_2 can be obtained as follows:

$$C_2 = \frac{y(T_s/2)}{2} e^{\frac{\omega T_s}{2}} \tag{6.6}$$

$$C_1 = C_2 e^{-\omega T_s} \tag{6.7}$$

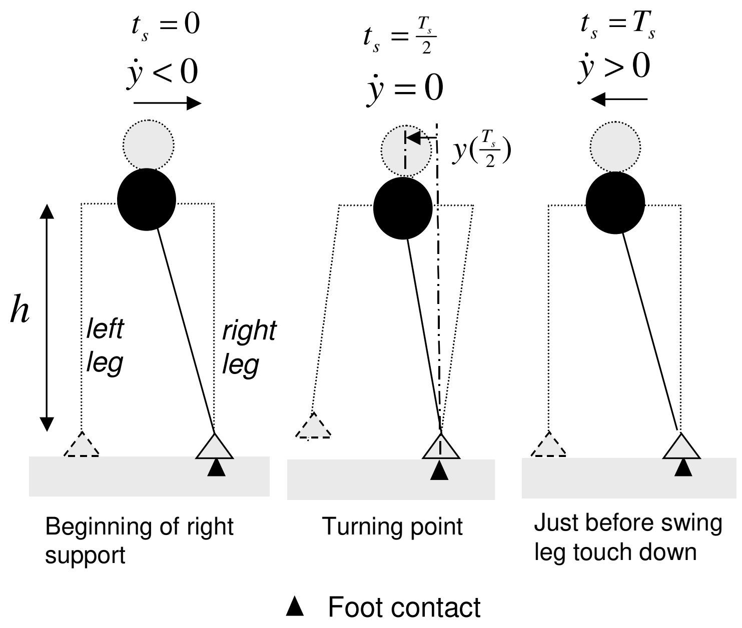


Figure 6-2: The frontal plane motion during the right support phase based on the linear inverted pendulum model. The black circle represents the point mass of the linear inverted pendulum model.

For this case, the control law at the stance roll ankle (for the right support phase) is given as follows:

$$\tau_{ar} = K_{ar}(y^d - y) + B_{ar}(\dot{y}^d - \dot{y}) \tag{6.8}$$

where y^d and $\dot{y}^d$ are the desired lateral position and velocity of the midpoint between the hips (measured from the stance ankle) given by the coarse model; K_{ar} and B_{ar} are constant gains.

The desired torque τ_{ar} from both control laws are bounded to prevent the stance foot from tipping over its side edges. Simple static analysis as in the sagittal plane is used to generate the bounds:

$$\tau_{aru} = K_{aru} M g l_{fw}/2 \tag{6.9}$$
$$\tau_{arl} = -K_{arl} M g l_{fw}/2 \tag{6.10}$$

where τ_{aru} and τ_{arl} are the upper and lower bounds for τ_{ar}; M is the total mass of the biped; g is the gravitational constant; l_{fw} is the foot width; K_{aru} and K_{arl} are the positive discount factors to compensate for any discrepancies due to unmodelled dynamics.

In the frontal plane, the reinforcement learning algorithm is used to learn the end position of the swing leg roll angle ϕ_f, whereas the swing time T_s is set by the sagittal plane implementation. The algorithm is summarized in Table 6.1. The effectiveness of both local control law candidates will be compared.

In the sagittal plane, the reinforcement learning implementation is almost the same as the implementation S_3. That is, a myopic agent is used. However, the swing time T_s is constant and the agent learns the horizontal end position of the swing ankle with reference to the hip x^f_{ha}. The swing leg strategy is as described in Subsection 5.2.2. The local control mechanism based on the stance ankle pitch joint is as described in Section 5.3. The description of this implementation is summarized in Table 6.2.

The algorithms for the frontal, sagittal and transverse planes are combined to form a three-dimensional (3D) walking algorithm. There are many ways to implement this algorithm. One approach is to train the sagittal plane's motion control first followed by the frontal plane's, or vice versa. Another approach is to implement them simultaneously. The latter approach is more attractive for the physical implementation since the former approaches require special training setup to constrain the biped to the sagittal or frontal plane motion. However, the latter approach requires proper partitioning of the failure conditions so that the cause of failure can be correctly attributed, that is, whether it is due to the sagittal or frontal plane motion. Both approaches assume that the dynamics in the sagittal plane and the frontal plane are weakly coupled.

The 3D algorithm is applied to the simulated M2. In every iteration, the biped starts to walk from a standing posture with both ankles directly below the respective hip joints. Its body is given an initial velocity of $[\dot{x}\,\dot{y}\,\dot{z}]^T = [0.6m/s\ -0.42m/s\,0m/s]^T$. The learning in both the sagittal and frontal planes is carried out simultaneously. The learning target is to achieve 100 seconds of walking without violating the failure conditions in both the sagittal and frontal planes. The desired walking speed is set at 0.4 m/s. The swing time T_s is fixed at 0.5 second. In the frontal plane, $y(T_s/2)$ is set to be equal to 1/8 of the hip spacing (0.184 m) for the local control based on Equation 6.8.

Figure 6-3 shows the resulting learning curves for the simulated M2. The solid and dotted graphs are the learning curves for the implementations in which the frontal plane local controls were based on Equation 6.5 and 6.8, respectively. The latter implementation has a much better performance.

From the data (Figure 6-4) of the frontal plane implementation after the learning target has been achieved, it is observed that the behavior of the biped in the frontal plane is "chaotic". That is, the action sequence $(u(n))$ from the frontal plane agent does not seem to converge to any regular pattern. It may be due to several reasons. Firstly, it may be the result of the approximated representation (using CMAC) of the Q-factors. It may also

Table 6.1: Reinforcement learning implementation for the frontal plane: F_1

Description	Remark
Implementation code	F_1
Swing time	constant(same value as the sagittal plane)
Learning output (action), u_2	End position of the swing leg roll angle, ϕ_f
Key parameters:	
Reward function	Equation 6.2.2
Θ_u	$0.26rad$
Θ_l	$-0.15rad$
C_u	0.1
C_l	-0.1
Discount factor γ_2	0.9 (farsighted)
Action set U_2	$\{0.001n \vert \text{ for } -0.06 \leq 0.001n \leq 0.1 \text{ and } n \in \mathbf{Z}\}$
Policy	modified ϵ-greedy ($\epsilon = 0$)
CMAC parameters	
Width of the receptive field for $\dot{y}^+$	$0.04m/s$
Width of the receptive field for ϕ^+	$0.04rad$
Width of the receptive field for θ^+	$0.04rad$
Width of the receptive field for x_{ha}^f	$0.04rad$
Number of the receptive fields layers C	64
Learning step-size α	0.125

Table 6.2: Reinforcement learning implementation for the sagittal plane: S_4

Description	Remark
Implementation code	S_4
Swing time, T_s	constant
Desired walking speed	constant
Learning output (action), u	Horizontal end position of the swing foot with reference to the hip x_{ha}^f
Key parameters:	
Reward function	Equation 5.5
V_u	$-0.1m/s$
V_l	$1.3m/s$
Z_l	$0.7m$
R_f	-1
K_r	0.5
Discount factor γ	0 (myopic)
Action set U	$\{0.02n \vert \text{for } -0.1 \leq 0.02n \leq 0.2 \text{ and } n \in \mathbf{Z}\}$
Policy	modified ϵ-greedy ($\epsilon = 0$)
CMAC	
Width of the receptive field for $\dot{x}^+$	$0.08m/s$
Width of the receptive field for x_{ha}^+	$0.064m$
Width of the receptive field for l_{step}	$0.16m$
Width of the receptive field for x_{ha}^f	$0.04m$
Number of the receptive fields layers, C	128
Learning step-size, α	0.1

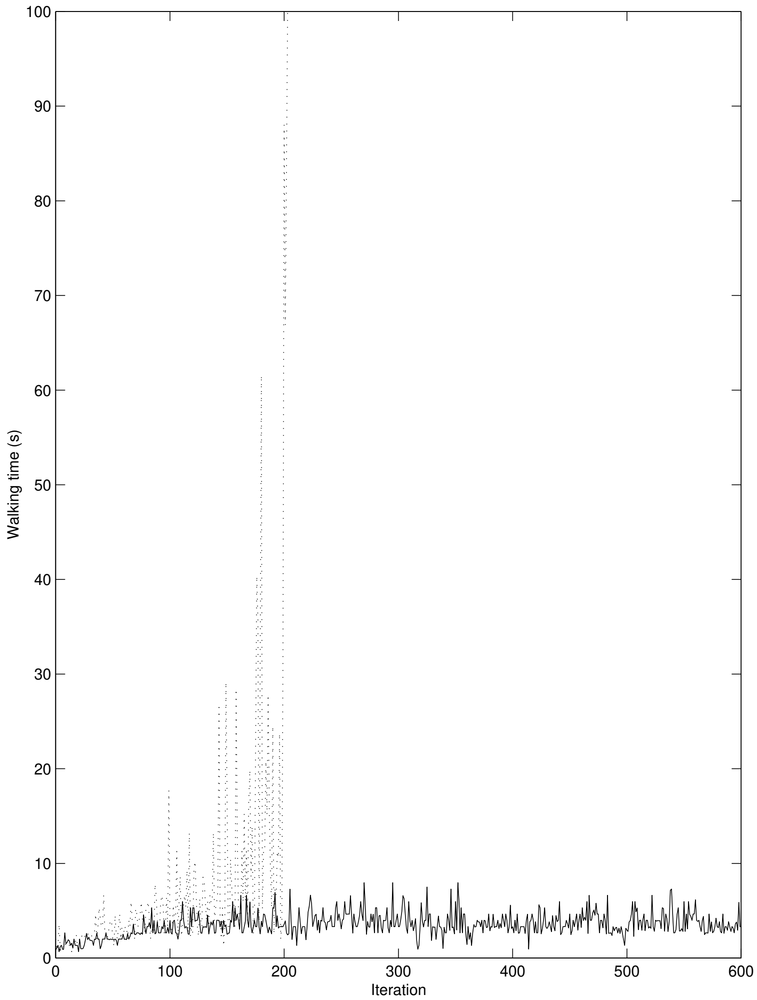


Figure 6-3: The learning curves for the simulated M2 when implementations F_1 and S_4 were used. The learning in both the sagittal and frontal planes was carried out simultaneously. The solid-line and dotted-line graphs are the learning curves for the implementations in which the frontal plane's local controls were based on Equations 6.5 and 6.8, respectively. The implementation whose frontal plane's local control was based on Equation 6.8 managed to achieve the learning target in about 200 learning iterations. The other implementation did not seem to improve its walking time even after around 600 learning iterations.

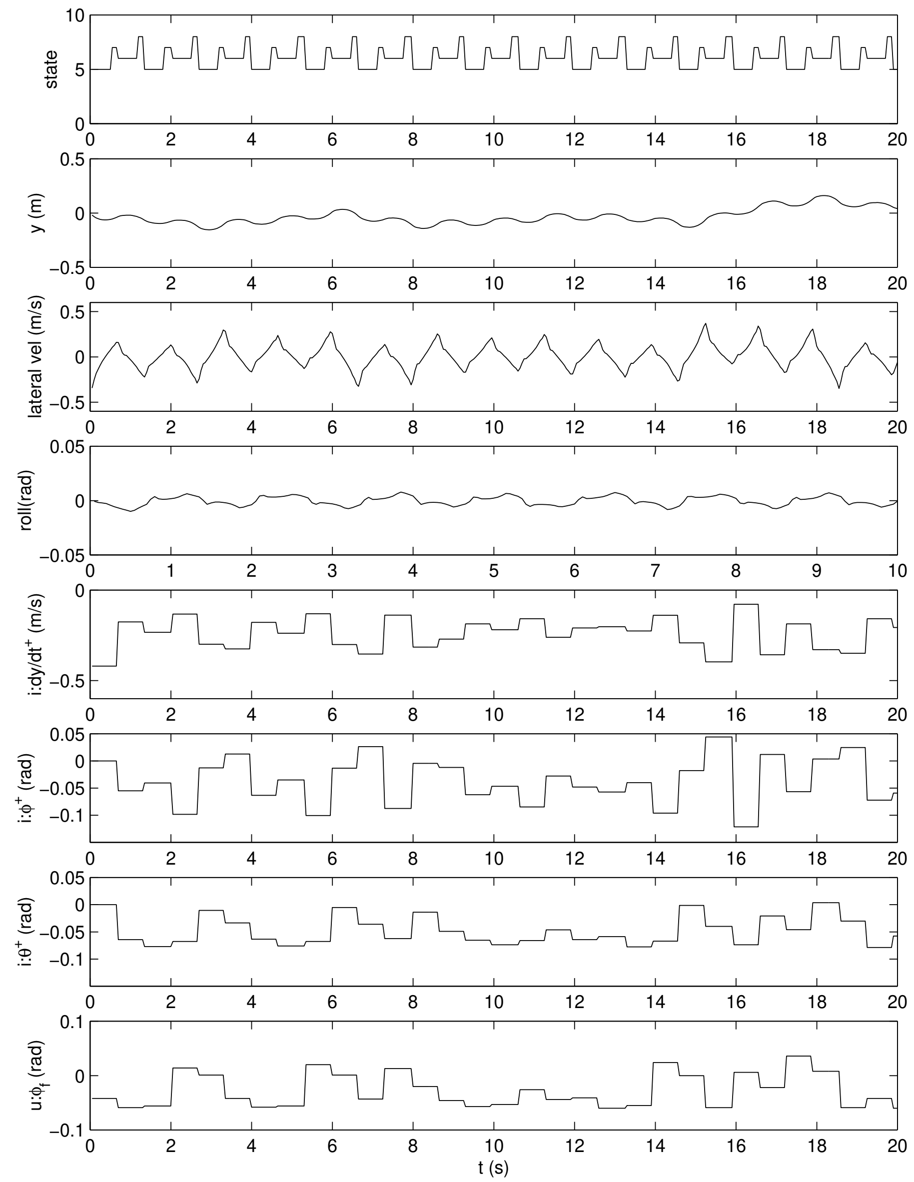


Figure 6-4: The simulation data (after the simulated M2 has achieved the learning target) for the implementation F_1 whose local control is based on Equation 6.8 .

be due to the lack of active exploration that could have helped to estimate the Q-factors better. It may also be due to the fact that there is no mechanism installed to make the agent behave in a regular manner. That is, the agent is just "told" to make sure the biped does not violate the failure conditions, but not to behave in a regular manner. The next section presents an implementation in which regularity for the frontal plane motion is built into the algorithm.

6.4 Notion of Symmetry

One key problem of the previous approach is that the frontal plane agent may behave in a chaotic manner. That is, the desired end position of the swing-hip roll angle may not converge to a limit point. One solution is to use a reward function that penalizes such a chaotic behavior. Another approach is to build regularity into the algorithm. The latter approach is adopted in this section and it is based on the notion of symmetry. Raibert [86] used it to design an algorithm for foot placement to control the forward speed of a hopping machine. His work resulted in a simple implementation which did not require any dynamic model of the legged machine though the dynamics of the machine were very complex to analyze.

The notion of symmetry is a simple but general concept. For the frontal plane motion, assuming that both legs have similar actuators' characteristics and inertia distribution, the swing leg roll angle ϕ of the biped is commanded to mirror the behavior of the stance leg roll angle θ. Assuming that the body roll angle is equal to zero at all times, the control law for the swing-hip roll joint (for the right suport phase) is formulated as follows:

$$\tau_{hrs} = K_{hrs}(\theta - \phi) + B_{hrs}(\dot{\theta} - \dot{\phi}) \tag{6.11}$$

where τ_{hrs} is the desired torque at the swing-hip roll joint; and K_{hrs} and B_{hrs} are the proportional and derivative gains, respectively. This control law can be viewed to be a rotational virtual spring-damper placed at the swing-hip roll joint.

The local control law is chosen to be the same as Equation 6.5. The desired torque τ_{ar} of the stance ankle roll joint is also bounded by the same bounds (Equations 6.9 and 6.10). A reinforcement learning algorithm almost similar to the implementation F_1 is used to learn the offset Δ for the desired swing-hip roll angle before the beginning of each step. That is, Equation 6.11 becomes $\tau_{hrs} = K_{hrs}(\theta + \Delta - \phi) + B_{hrs}(\dot{\theta} - \dot{\phi})$. The details of this RL implementation are not listed here because it turns out that learning is not required for such an implementation.

In the sagittal plane, implementation S_4 (Table 6.2) is adopted. That is, a myopic agent is used. The agent learns the horizontal end position of the swing ankle with reference to the hip x^f_{ha}. The swing leg strategy is as described in Subsection 5.2.2. The local control mechanism based on the stance ankle pitch joint is as described in Section 5.3.

The algorithms for the frontal, sagittal and transverse planes are combined to form a three-dimensional (3D) walking algorithm which is again applied to the simulated M2. In every iteration, the biped starts to walk from a standing posture. Its body is given an initial velocity of $[\dot{x}\,\dot{y}\,\dot{z}]^T = [0.6m/s\ -0.37m/s\,0m/s]^T$. The learning in both the sagittal and frontal planes is carried out simultaneously. The learning target is to achieve 100 seconds of walking without violating the failure conditions in both the sagittal and frontal planes. The desired walking speed and height are set at 0.4 m/s and 0.84 m, respectively. The swing time is fixed at 0.5 second.

The simulation result reveals that the frontal plane learning is not required for the 3D walking. That is, the swing-hip roll joint control based on Equation 6.11 and the local control law based on Equation 6.5 for the stance ankle roll joint yield a stable frontal plane motion (provided a right set of parameter values are chosen). A learning curve for such an implementation is shown in Figure 6-5. The learning curve shows that the biped achieved 100 seconds of walking within 50 iterations. It was much faster than the learning curve in the previous implementation because the biped only encountered failures in the sagittal plane. Thus, the learning curve is comparable to that of the planar (sagittal plane) implementation.

Figure 6-6 shows the stick diagram of the dynamic walking of the simulated M2 after the learning target was achieved. The body trajectory of the biped are shown in Figure 6-7. In the sagittal plane, the forward velocity and the pitch angle of the body were well-behaved. The average body height (measured from the ground to the midpoint of the hips) was below the desired value (0.84m/s). This was mainly because the DC component of the vertical virtual component was not high enough. To solve this problem, the DC component can be increased. Alternatively, a robust adaptive control approach can be adopted as in [19] for the height regulation.

In the frontal plane, the lateral velocity and the roll angle of the body were well-behaved. However, there was a slight drifting of the horizontal position y of the body. It is mainly because there is an asymmetry in the frontal plane motion of the biped. It is evident from the legs' roll angles plots (ϕ and θ) in Figure 6-8 that both angles were different between the right and left legs. This could be due to the inherent dynamics in the frontal plane that depends on the initial conditions. It could also be due to the fact that the sagittal plane motion has some effects on the frontal plane motion. And since the the sagittal plane data like the step length and the desired end position of the swing ankle x_{ha}^{f} did not converge to a limit point (see the last two graphs of Figure 6-8), the frontal plane motion would also be influenced. Of course, the influence can be both ways too.

There are several ways to correct the drifting in the lateral direction. One of which is to create asymmetry in the local control at the stance ankle roll joint when the drifting is exceeded by a preset amount. Another approach is to use the transverse motion (yaw). For example, if the biped has drifted to one side, the stance hip yaw joint can be used to change the walking direction to nullify the drift.

The feasibility of achieving the frontal plane control based on the notion of symmetry and the local control mechanism without the need for learning is a very interesting finding of this implementation. In fact, the frontal plane control strategy is quite "robust" because the biped did not encounter a single failure in the frontal plane even while it was learning the sagittal plane motion.

Note that the notion of symmetry does not by itself produces the solution. However, it can be used to derive candidate strategies for the motion control. Once a candidate strategy is created, the architecture proposed in this thesis acts as a potential testbed for it. In this case, the strategy for the frontal plane motion just happens to yield a positive outcome even without any learning.

The following subsection analyzes the frontal plane motion strategy adopted in this section using a simple model. The model is meant to provide some insight for the strategy, especially with regard to the stabilization issue.

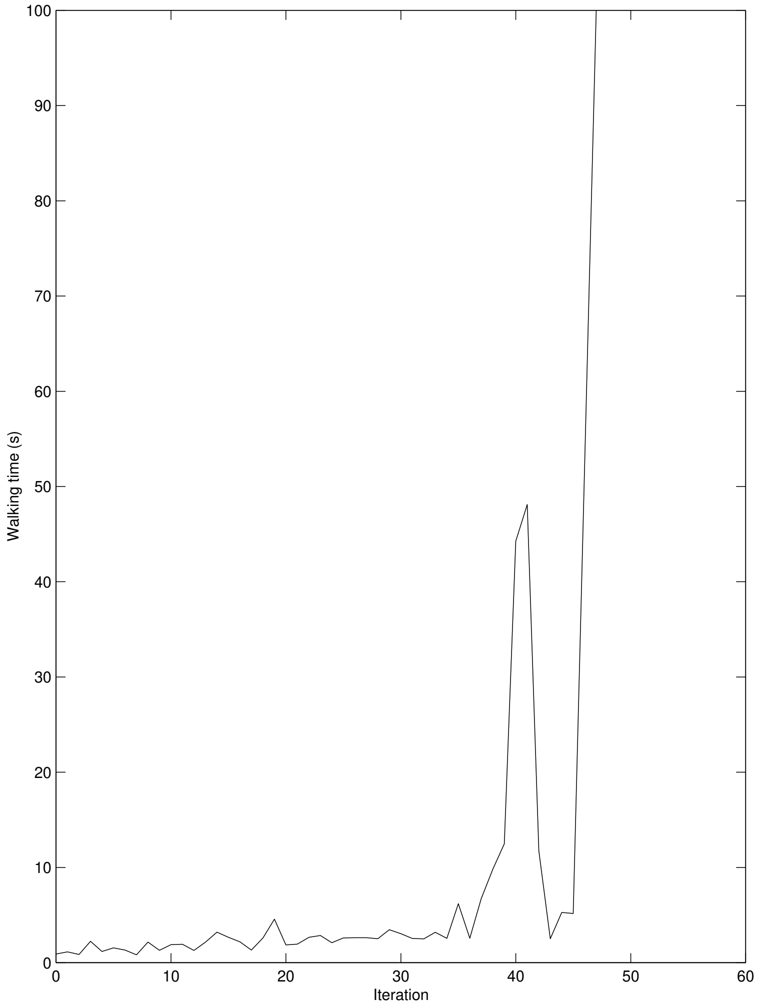


Figure 6-5: A learning curve for the dynamic walking of the simulated M2. The frontal plane implementation was based on the notion of symmetry. The sagittal plane implementation was S_4.

6.4.1 Analysis

This subsection describes the analysis of the symmetry approach for the lateral balance. A simple linear double pendulum model as shown in Figure 6-9 is used in the analysis to validate that this approach can result in a stable motion. In this model, the body is a point

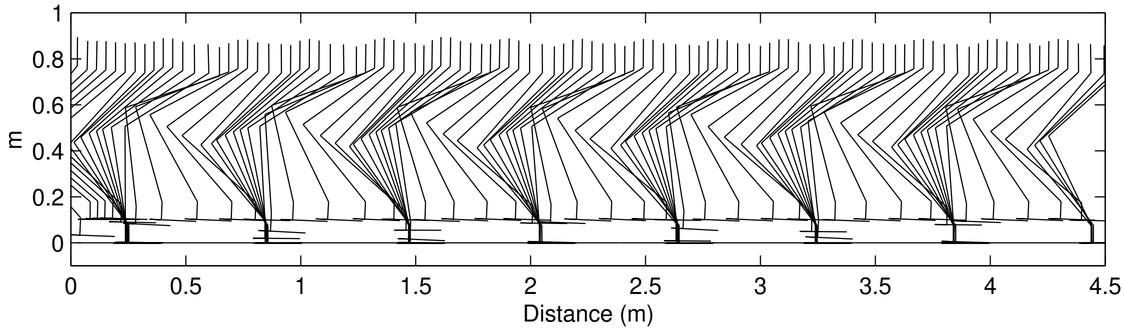


Figure 6-6: Stick diagram of the dynamic walking of the simulated M2 after the learning target was achieved. Only the left leg is shown in the diagram and the images are 0.1 second apart.

mass M that is constrained to move along the horizontal constraint line 1. The swing leg's mass m is assumed to be concentrated at the foot and it is constrained to move along the horizontal constraint line 2. x_1 and x_2 are chosen to be the generalized coordinates for this model. x_1 is the horizontal distance of the body mass M measured from the stance ankle. x_2 is the horizontal distance of the leg mass m measured from the body mass. τ_1 is the torque exerted on the stance leg by the ground. τ_2 is the torque exerted on the swing leg by the stance leg. Let's assume that the biped's single support period or swing time T_s is fixed, and the support exchange occurs instantaneously.

If the support exchange event is temporarily neglected, the dynamic equations of this model can be obtained by Newtonian mechanics as follows:

$$\begin{bmatrix} (m+M)h & mh \\ mh & mh \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} + \begin{bmatrix} -(m+M)g & 0 \\ 0 & mg \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{6.12}$$

From the symmetry approach adopted earlier, the control laws for τ_1 and τ_2 are formulated as follows:

$$\tau_1 = K_1 \dot{x}_1 \tag{6.13}$$

$$\tau_2 = K_P(x_1 - x_2) + K_D(\dot{x}_1 - \dot{x}_2) \tag{6.14}$$

Now, let's substitute the control laws for τ_1 and τ_2 into Equation 6.12 to yield Equation 6.15:

$$\mathbf{A} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} + \mathbf{B} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \mathbf{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.15}$$

where

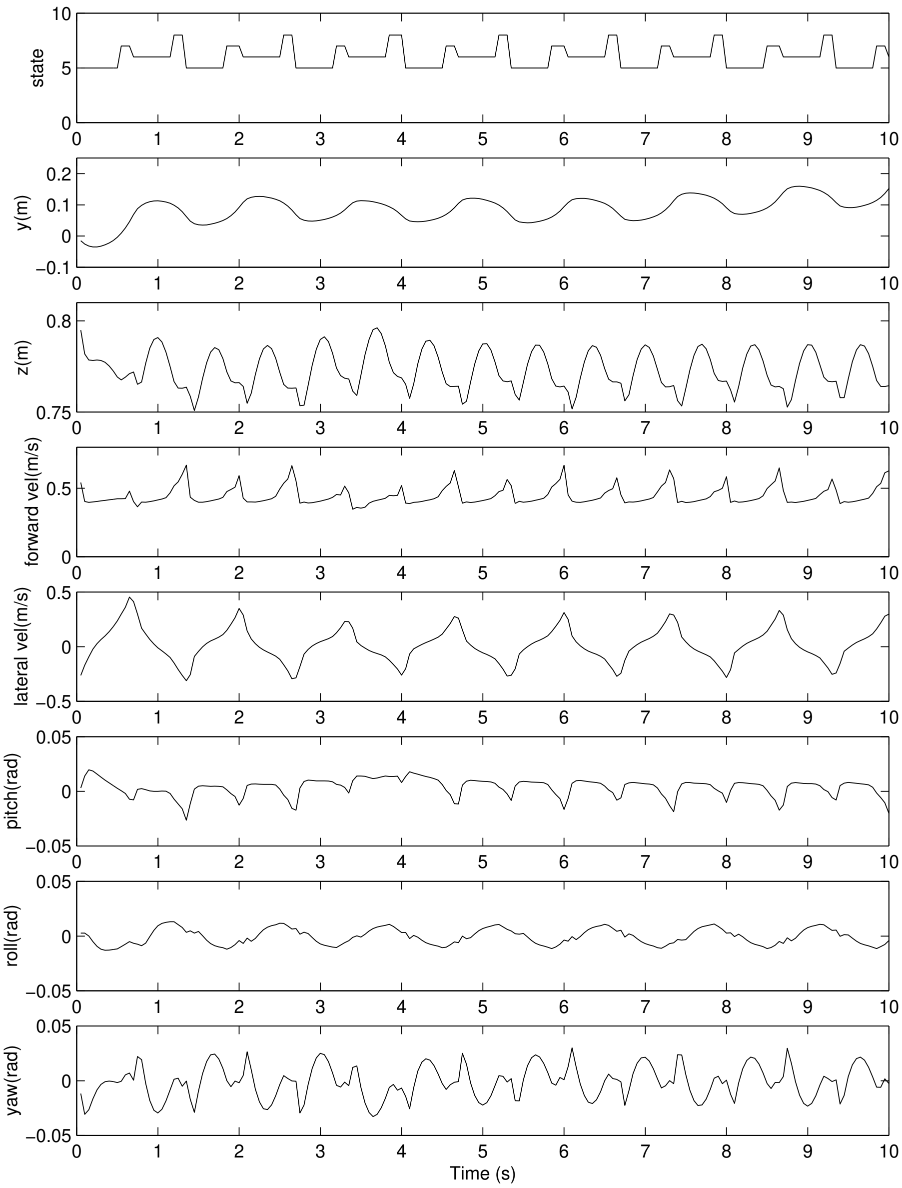


Figure 6-7: The body trajectory of M2 during the dynamic walking after the learning target was achieved. The frontal plane implementation was based on the notion of symmetry. The sagittal plane implementation was S_4.

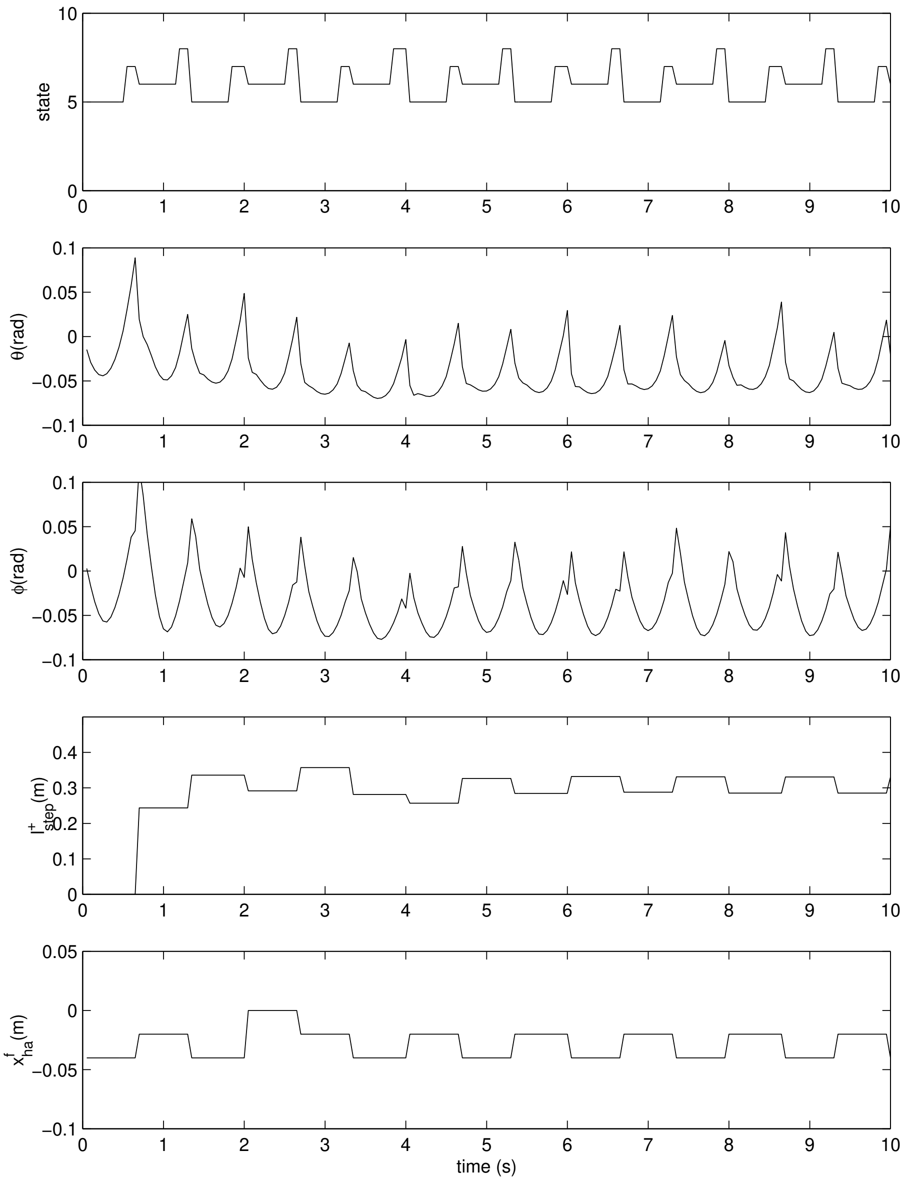


Figure 6-8: The legs' roll angle trajectories of M2 during the dynamic walking. The frontal plane implementation was based on the notion of symmetry. The sagittal plane implementation was S_4. The present step length l^+_{step} and action sequences of the sagittal plane implementation are also included.

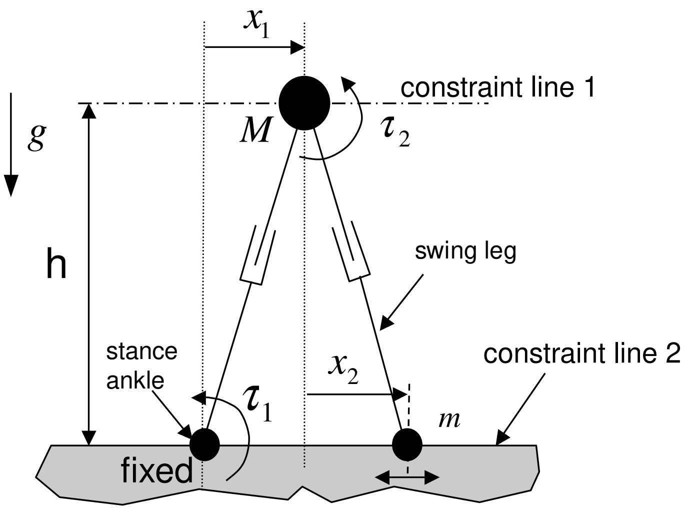


Figure 6-9: A linear double pendulum model for the analysis of the symmetry approach used in the frontal plane algorithm.

$$\mathbf{A} = \begin{bmatrix} (m+M)h & mh \\ mh & mh \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} -K_D + K_1 & K_D \\ -K_D & K_D \end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix} -(m+M)g - K_P & K_P \\ -K_P & mg + K_P \end{bmatrix}$$

Equation 6.15 can be rearranged as follows:

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \mathbf{D} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \mathbf{E} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{6.16}$$

where

$$\mathbf{D} = -\mathbf{A}^{-1}\mathbf{B}$$
$$\mathbf{E} = -\mathbf{A}^{-1}\mathbf{C}$$

Finally, the above equations can be expressed in a state space form:

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ E(1,1) & D(1,1) & E(1,2) & D(1,2) \\ 0 & 0 & 0 & 1 \\ E(2,1) & D(2,1) & E(2,2) & D(2,2) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{6.17}$$

where $\vec{y} = [y_1\, y_2\, y_3\, y_4]^T = [x_1\, \dot{x}_1\, x_2\, \dot{x}_2]^T$.

Let $\vec{y}_i^{(n)}$ denote the initial state vector (subscript i) at the beginning of the n th step. If the swing time T_s is fixed, the final state vector (subscript f) just before the support exchange is given by the explicit solution of Equation 6.4.1 as follows:

$$\vec{y}_f^{(n)} = \exp(\mathbf{F}T_s)\vec{y}_i^{(n)} \tag{6.18}$$

where

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ E(1,1) & D(1,1) & E(1,2) & D(1,2) \\ 0 & 0 & 0 & 1 \\ E(2,1) & D(2,1) & E(2,2) & D(2,2) \end{bmatrix}$$

Now, let's consider the support exchange event. Let's assume that the support exchange is instantaneous and the previous stance foot is not acted on by any impulsive force. Also, by the conservation of angular momentum about the ankle of the new supporting leg and assuming that both masses remain in their respective constraint lines, the following boundary conditions can be obtained:

$$\vec{y}_i^{(n+1)} = \mathbf{G}\vec{y}_f^{(n)} \tag{6.19}$$

where

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

By substituting Equation 6.19 into Equation 6.18, the following discrete LTI system is obtained:

$$\vec{y}_i^{(n+1)} = \mathbf{H}\vec{y}_i^{(n)} \tag{6.20}$$

where $\mathbf{H} = \mathbf{G}\exp(\mathbf{F}T_s)$ is the state transition matrix.

From linear control theory, the modulus of all the eigenvalues λ_i of $\mathbf{H}$ must be less than unity for Equation 6.20 to be asymptotically stable. Now, let's set the values for the parameters in this model so that quantitative analysis can be carried out. The key parameters are set as follows: $M = 20kg$, $m = 2kg$, $h = 0.84m$, $T_s = 0.5s$, and $g = 9.81m/s^2$. The maximum modulus of the eigenvalues is plotted against K_P and K_D for $K_1 = 1, 10, 20$ and 50 using $Matlab^{TM}$ as shown in Figure 6-10. The figure shows that the system indeed has a zone in the K_P-K_D plane where the maximum modulus of the eigenvalues is less than one. The zone is especially obvious and large for $K_1 = 10$. From the figure, it is also observed that K_P and K_D should not be too large or too small. Having large values for these parameters means better tracking of the desired trajectory. That is, perfect symmetry is not desirable for the algorithm. On the other hand, if K_P and K_D are too small, this will result in a poor tracking of the desired trajectory. That is, the stance leg roll angle has little influence on the swing leg roll angle.

This analysis validates the frontal plane algorithm presented in this section. A stable motion in the frontal plane can be achieved if a proper set of K_P, K_D and K_1 is chosen.

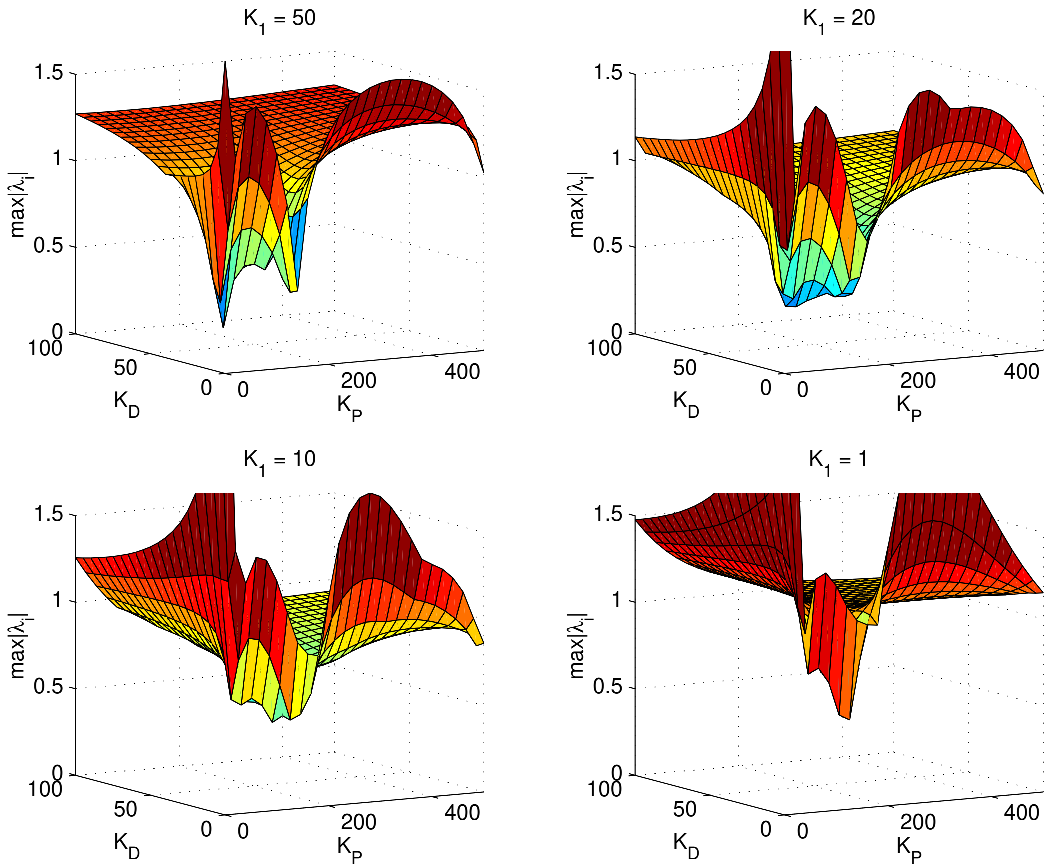


Figure 6-10: Four surface plots of the maximum modulus of the eigenvalues of **H** against K_P and K_D corresponding to $K_1 = 1, 10, 20$ and 50.

6.5 Summary

In this chapter, the 3D walking algorithms were constructed based on the assumption that the motions in the sagittal, frontal and transverse planes could be independently considered. The successful implementations of the algorithms indirectly validated this assumption and the proposed control architecture.

In Section 6.3, the reinforcement learning algorithm was used to learn the end position of the swing leg roll angle ϕ_f before the beginning of each single support phase. Two local control law candidates for the stance ankle roll joint were compared. The simulation results demonstrate that the choice of the local control law for the stance ankle roll joint can affect the learning rate.

Section 6.4 illustrated how one could tap the hidden potential of the notion of symmetry for the bipedal walking task. The successful application of the notion for the frontal plane motion control without complex control laws endorses Raibert's claim [86] that symmetry could be utilized to simplify the control of legged robots. In this thesis, the strategy was found during an implementation of the proposed architecture. That is, the process of implementing an algorithm based on the proposed architecture may result in a new control approach for bipedal locomotion. The learning rates for the constructed algorithms can be used to gauge their relative effectiveness.

The simulation results in both Section 6.3 and 6.4 also reveal that the choice of the control law for the stance ankle depends on the selected swing leg strategy. The control law that does not work well for the frontal plane implementation in Section 6.3 actually works well for the implementation in Section 6.4.

The next chapter gives further examples of how the algorithms developed thus far can be modified to generate other behaviors of walking. One behavior is variable-speed walking. This is a critical but difficult behavior that one would like a bipedal robot to possess. The algorithms will also be modified so that the simulated M2 can achieve a "man-like" walking posture.

Chapter 7

Other Implementations

Variable-speed walking is one of the desired performance features of bipedal walking, as mentioned in Section 3.2. This chapter includes the implementation in which the simulated M2 learns to achieve such a behavior. The variable speed walking is implemented by simply augmenting the state space of the sagittal plane learning algorithm.

To demonstrate the flexibility of the proposed control architecture, a human-like walking implementation is included. In the implementation, the swing leg strategy and the height control algorithm is modified to illustrate the freedom of algorithm design.

7.1 Variable-Speed Walking

So far, the simulation analyses have been conducted for constant desired walking speed. This section demonstrates how the control algorithms constructed in earlier chapters can be extended for variable-speed walking. Variable-speed walking is a desirable performance requirement for bipedal walking. For example, the walking speed may need to be changed depending on environmental conditions. In a crowded environment, it may be desirable for the biped to walk at a slower speed so as to reduce the chances of running into other objects or human beings. In an open space, the biped may walk at a higher speed so as to reduce the travelling time.

The variable-speed walking behavior is usually hard to achieve because it involves transient walking phases (as compared to steady walking). In this thesis, the algorithm for such a behavior is again partitioned into sub-algorithms corresponding to the sagittal, frontal and transverse planes.

The control algorithms for the frontal and transverse planes are the same as before. For the frontal plane, the control algorithm is as described in Section 6.2. No learning algorithm is applied. Lateral balance is achieved by the symmetry approach as in Section 6.4. For the transverse plane, the control algorithm is as described in Section 6.2.

In the sagittal plane, only a myopic agent is applicable for the learning implementation. A farsighted implementation is not applicable simply because there is an extra input to the system at each decision time step. The input is the desired walking speed $\dot{x}_d$ and it is exogenous to the biped system. That is, the biped does not have any choice over this input. Thus, at the beginning of each time step, given the desired walking speed, the biped needs to decide what action to take in order to satisfy the desired walking speed and to determine how good the executed action is after a reward value is obtained. It is not feasible to look at the cumulative reward (return) here because the profile of the desired walking speed may affect the cumulative reward.

The sagittal plane learning implementation is almost the same as the implementation S_4 (see Table 6.2). That is, a myopic agent is adopted and the action is the horizontal end position of the swing leg with reference to the hip x_{ha}^f. The state variables are the same as before. However, the desired walking speed $\dot{x}_d$ is considered to be an input to the reinforcement learning algorithm. That is, given the state variables and the desired walking speed, the agent has to select an action that results in a good behavior. The desired walking speed may change at the next step.

The algorithms for the frontal, sagittal and transverse planes are again combined to form a three-dimensional (3D) walking algorithm. The algorithm is applied to the simulated M2. In every iteration, the biped starts to walk from a standing posture. Its body is given an initial velocity of $[\dot{x}\,\dot{y}\,\dot{z}]^T = [\dot{x}_i\ -0.37m/s\,0m/s]^T$, where $\dot{x}_i$ is a uniformly distributed random variable between 0.2 and $0.7m/s$. To train the biped so that it can achieve variable-speed walking, the desired walking speed $\dot{x}_d$ is varied every six seconds based on the same uniform random distribution as $\dot{x}_i$. Note that, the desired walking speed is only updated at the next support exchange event after the six seconds have lapsed.

A learning curve for the implementation is shown in Figure 7-1. It is observed that the number of trials taken to achieve the learning target ($100s$ of continuous walking) is greater than that of the constant desired walking speed case (Figure 6-5). This is because the learning task is now more complex.

The body trajectory of the biped during the variable-speed walking is shown in Figure 7-2. The lateral velocity is well-behaved. This demonstrates the effectiveness of lateral balance by the notion of symmetry.

The forward velocity is also well-behaved even though there are step changes in the desired velocity. However, for the first six seconds of the graph, the peak of the forward-velocity profile seems to behavior like a "period-2" cycle [100], evident from the step length plot (bottom graph of Figure 7-2). This could be due to the fact that there is no mechanism to penalize such a behavior, and, since the overall algorithm is nonlinear, it is not surprising to encounter such a behavior.

7.2 Human-like Walking

So far, the proposed control architecture has only been applied to a bird-like walking posture. In this section, the walking algorithm will be modified to achieve a 3D human-like walking posture. Human-like walking is defined to be that in which the stance knee does not point toward the rear at any instant. Instead, it points toward the front during part of or the whole period of the walking cycle.

In this implementation, mechanical limits (see Figure 7-3) are assumed to exist at the knee joints to prevent the legs from hyper-extension (i.e., thigh and shank being aligned). The minimum knee angle θ_k is set to 10 degrees. To maintain the walking height, the stance knee joint is servoed based on a PD controller (or virtual rotational spring-damper pair) to

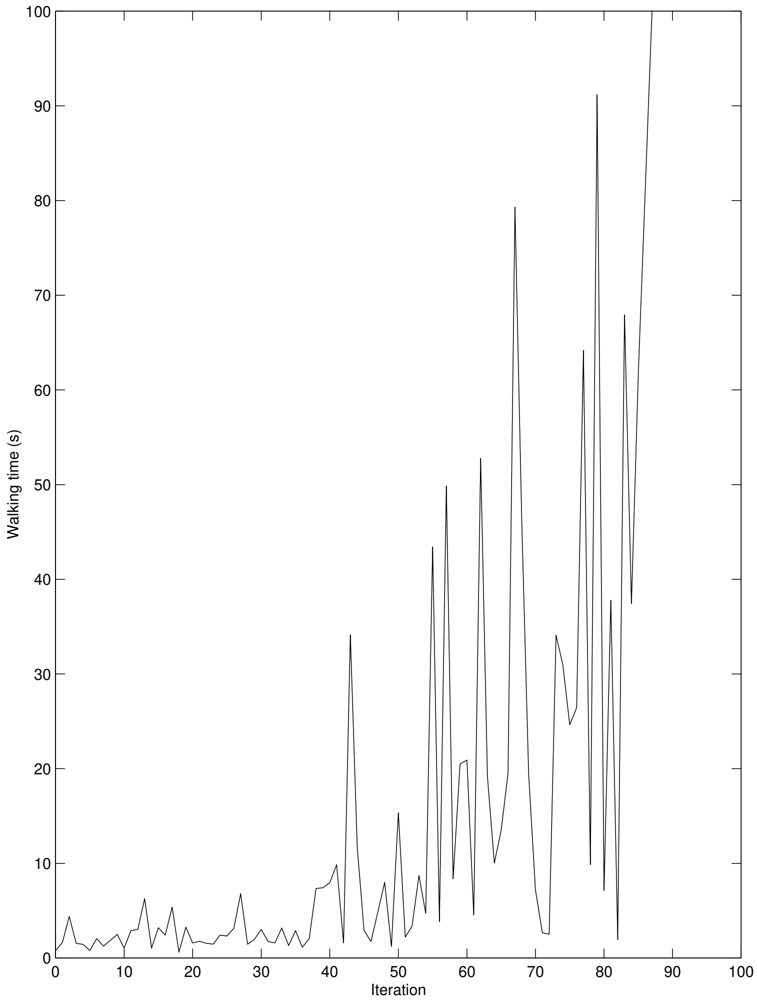


Figure 7-1: A learning curve for the variable-speed walking implementation.

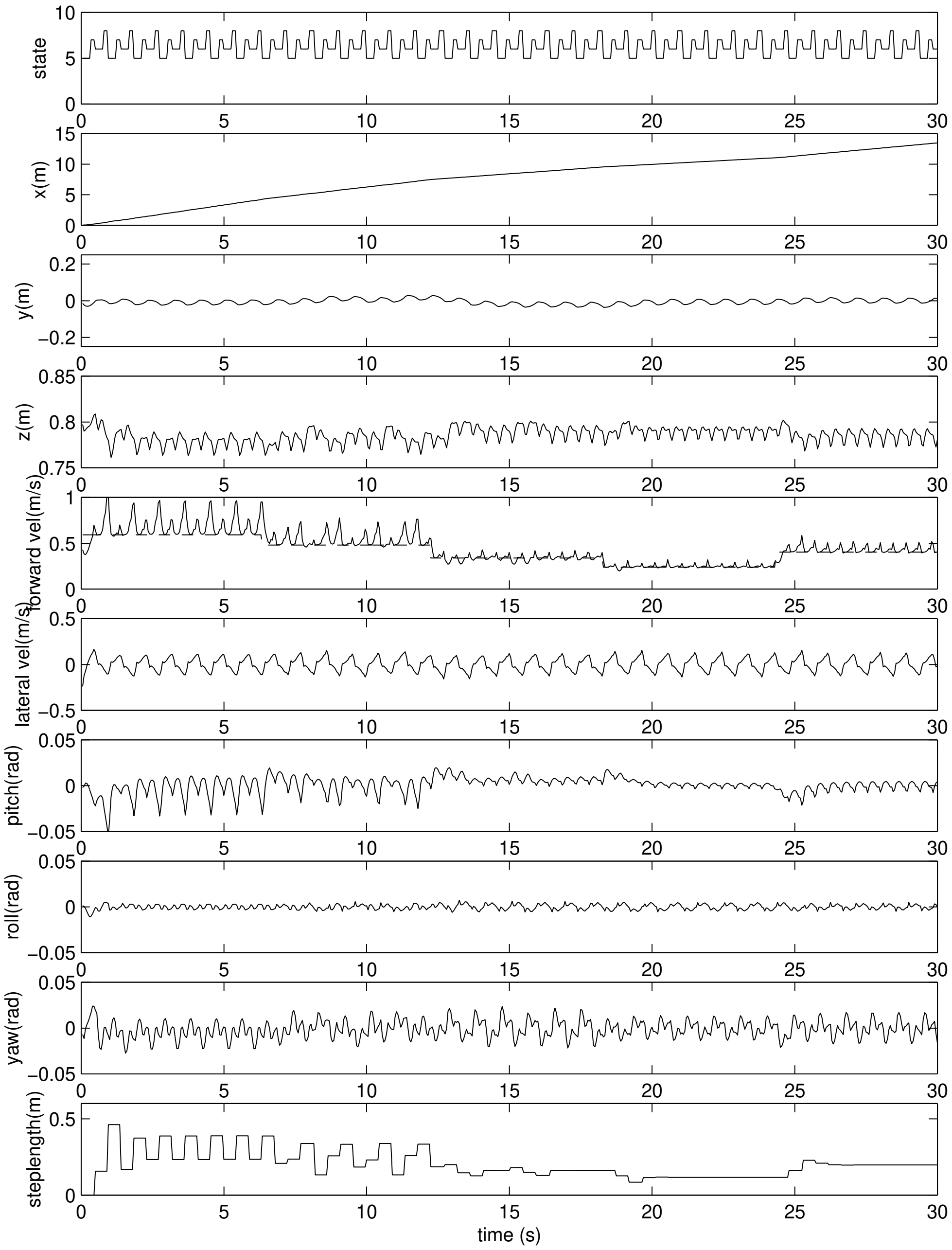


Figure 7-2: The body trajectory and selected output variables of the simulated M2 during the variable-speed walking.

a value (say 0 degrees) which is less than the minimum knee angle. This results in the stance knee joint touching the mechanical limit and staying at this position throughout the stance phase. This height control approach is similar to that used in [82]. The stance-ankle pitch joint and the stance-hip pitch joint are used in ways similar to implementation S_4. The resulting walking posture is a slightly bent-knee posture like that of the Honda humanoid robots.

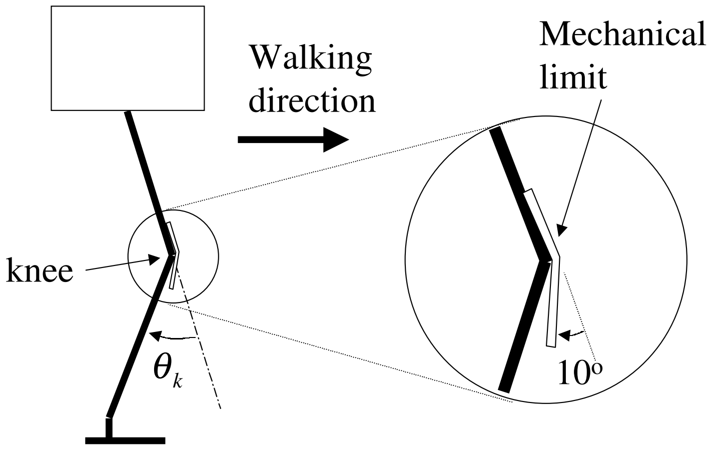


Figure 7-3: Mechanical limit is used to prevent the legs from hyper-extension in human-like walking simulations.

The swing leg strategy in the sagittal plane is deliberately modified from that used in the bird-like walking implementations. In the first part of the swinging phase (say 1/10 of the total swing time), the swing hip's roll joint is given a negative square pulse input while the swing knee is limp so that the natural dynamics cause the knee to further bend toward the positive direction (similar to [82]). After that, the desired trajectories are fitted by two third-degree polynomials (one for the vertical direction and one for the horizontal direction) as before, with the initial points provided by the detected state of the swing leg after the square pulse at the hip. The end position of the swing ankle in the x-direction (with respect to the hip) is provided by the reinforcement learning algorithm similar to implementation S_4. The rest of the swing leg implementation is the same as that in the bird-like walking algorithm. The frontal and transverse plane implementations are similar to those used in Section 6.4 and 7.1.

The combined algorithm is then applied to the simulated M2 with the position limits at the knee joints. The desired forward velocity is set at $0.7m/s$. A learning curve for the implementation is as shown in Figure 7-4. The learning rate is comparable to the result of the bird-like walking implementation (Section 6.4). A stick diagram of the human-like walking is presented in Figure 7-5. The success of the implementation illustrates that the proposed control architecture is flexible and general.

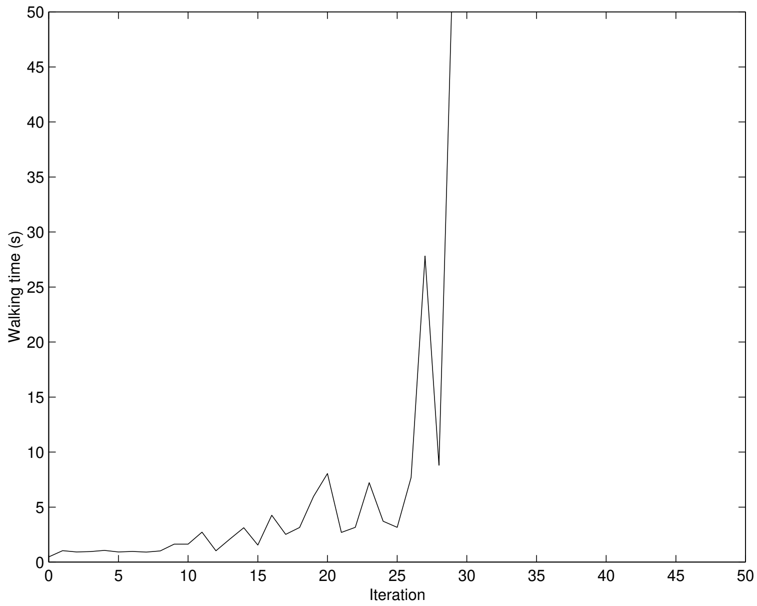


Figure 7-4: A learning curve for the human-like walking implementation.

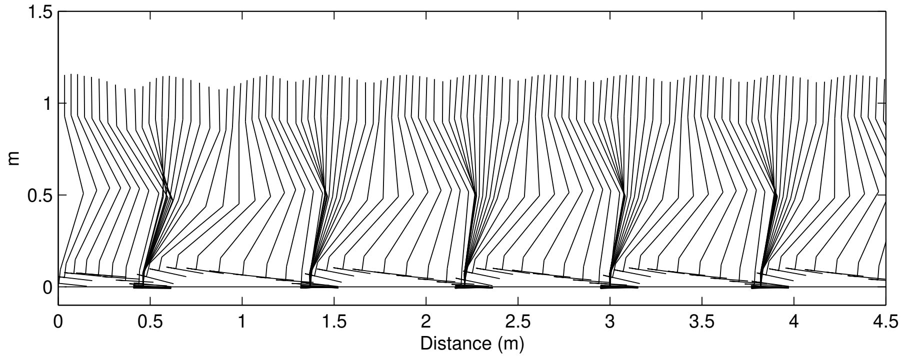


Figure 7-5: Stick diagram of the “human-like” walking of the simulated M2. Only the left leg is shown in the diagram and the images are 0.05 second apart.

Chapter 8

Conclusions

Based on a "divide-and-conquer" approach, the 3D dynamic bipedal walking control task is partitioned into sub-tasks in three orthogonal planes: sagittal, frontal and transverse. The learning algorithm is only applied to those sub-tasks that do not have any simple or obvious solution. This thesis demonstrates by simulation analysis that dynamic bipedal walking can be achieved by learning the key parameters of the swing leg. The Q-learning algorithm using CMAC as the function approximator is selected to be the learning method in all the implementations. The resulting dynamic walking algorithms do not require a complex dynamic model. The learning rates of the learning algorithms are quite high. Thus, the algorithms should be applicable to a physical robot. The simulation results also validate that the 3D walking task can be handled by breaking it down into three 2D motion control sub-tasks corresponding to each of the three orthogonal planes.

In previous research, a set of nominal joint trajectories was applied directly to the biped without considering the torque limitations at the ankle joints. This usually necessitated a big foot base in order to achieve the trajectory tracking without causing any instability to the stance ankle joint. On the contrary, this thesis considers the torque limitations of the stance ankle joints from the beginning based on the dimensions of the feet and the robot's weight. Once the control laws for the stance ankle joints have been set, the learning algorithms simply learn the appropriate swing leg parameters that result in a stable walking gait. The simulation results have demonstrated that even with the torque limitations, the stance ankle joints can drastically increase the learning rate in both the sagittal and frontal planes. They have also demonstrated that if the control laws for the stance ankle joints are not appropriate, the resulting learning system may not achieve the learning target at all. The choices of the control laws for the stance ankle joints depend on the swing leg strategy.

In the frontal plane, the notion of symmetry is applied in one of the algorithms. With a proper local control law at the stance ankle joint and a proper set of parameter values, the algorithm results in stable lateral balance even without any learning. A dynamic model is constructed to validate the approach.

The proposed control architecture is general in that it can be applied, with minimal parameter tunings, across different bipeds that have different actuator behaviors, inertia, or length parameters. Each of the bipeds simply needs to go through its respective training cycle. It is also flexible enough to be applied to different walking postures, for example, "bird-like" and "human-like" walking postures.

The learning part of the control algorithm can be used to evaluate indirectly various strategies for walking. For example, by examining the learning performance, the effectiveness of a local control law at the stance ankle joint can be judged. Various swing leg strategies may also be applied to see their relative effectiveness based on the respective learning rates.

Although the reinforcement learning method is only treated as a tool in this thesis, the success of the implementations depends on the behavior of the reinforcement learning algorithm. In the learning implementation, the Q-learning algorithm using CMAC for the Q-factors approximation is used. The algorithm allows the choice between "myopic" and "farsighted" learning agents, selected by simply setting the value of a parameter (γ). For a "farsighted" learning agent, the reward function needs not be very precise. This is good for an initial implementation in which minimum knowledge is available. If a precise reward function that correctly evaluates the action is available, a "myopic" learning agent usually results in better performance. In fact, the choice of the reward function is an art.

Although active exploration is desirable for better estimation of the Q-factors, it is not practical for a control problem. Even without active exploration of the state-action space, the learning target of the learning algorithm can still be achieved. This is only possible if the learning agent does not require accurate estimation of the Q-factors to achieve the learning target.

8.1 Future work

Several questions are generated in this research which the author hopes to answer or solve in the future. One question is on the robustness of the walking behavior after the learning target has been achieved. So far, no robustness measure for dynamic bipedal walking has been proposed. Thus, an important task is to search for such a measure before one could discuss the robustness of a bipedal walking behavior. Alternatively, the robustness can be indirectly checked by perturbing the walking behavior, for example, by applying an external force to the biped. The robot can also be made to walk on some randomly (but smoothly) varying terrain, or on different ground materials/types (for example, on sand versus on cement). However, the learned behavior may not be robust for those circumstances that are not encountered in the learning process. This is an intrinsic property of learning by trial-and-error. Such a property can also be observed in the biological counterparts. For example, a child who has first learned to walk successfully on a hard surface will most likely fall when he/she first steps onto sand.

The robustness of the system may be improved by adding other local control mechanisms. For example, to improve the forward velocity control robustness, one possibility is to add the local control mechanism based on the body pitching action. Using this mechanism, the robot can shift its body's center of gravity forwards or backwards to assist in the forward velocity control.

There is also a question of whether to turn off the learning after the learning target has been achieved. There may be a learning interference problem if the learning is not turned off and more disturbances or new situations are encountered later. This phenomenon is a typical problem in machine learning and is part of ongoing research in the field.

This research also opens up other possibilities with regards to the implementation. It would be interesting to study the feasibility of learning both the time and the spatial parameters of the swing leg control simultaneously, instead of just one at a time. Having more choices usually means more flexibility but also increases the complexity of the problem.

Since the ability to walk on rough terrain is touted to be an advantage of legged locomotion, rough terrain implementation is a natural extension of the algorithms developed in this thesis. It may be possible to extend the level ground algorithms for rough terrain locomotion based on just geometric consideration as in [20]. Alternatively, the learning algorithms can be augmented by including additional state variables, for example, detected slope gradient. In any case, the biped may need to be trained by walking on a terrain which has a rich set of slope variations to ensure that the biped is exposed to various situations and able to gather information about them.

Besides the rough terrain locomotion, the biped should also have the capability to change its walking direction. Honda's humanoid robot has demonstrated such a capability. However, it is achieved by a quasi-static turning strategy. The translational velocity of the body is zero while the robot is stepping and changing the yaw angle. It would be interesting to generate an algorithm that allows the biped to turn dynamically while the forward walking speed is being maintained.

The success of reinforcement learning depends very much on the reward function. For bipedal walking problems like dynamic walking and turning, due to their lack of well-defined reward functions, it is difficult to apply reinforcement learning to them. Even for the forward walking task, the chosen reward function for the myopic learning agent in the sagittal plane implementation is not perfect. For example, it does not penalize the multiple-period stepping cycle or even chaotic walking behaviors. Therefore, it is critical to explore more reward functions in the future especially if more complex walking behaviors are to be achieved.

In this thesis, the learning algorithms are used to learn the key parameters of the swing leg control. The key parameters are indirectly obtained by searching an intermediate function (Q-factors). It would be interesting to apply *reverse engineering* to learn the direct mapping from the state variables to the key parameters after the learning target has been achieved. The learned behavior may be used to train a general function approximator (for example, a feedforward neural network) or to obtain the set of parameter values of simple control laws (for example, affine function of the state variables).

Appendix A

Transformation Equation

This appendix includes the derivation of the transformation equation used for the stance leg control during the single support phase (in the sagittal plane) [81]. The equation is used to generate the hip pitch and knee torques for the stance leg given the virtual forces generated by the virtual components. The ankle pitch torque is assumed to be zero in the derivation.

Figure A-1 shows a planar four-link, three-joint model which is used to represent the stance leg of biped. Coordinate frame {A} is attached to the foot, and frame {B} is attached to the body. The angles θ_a, θ_k, and θ_h are those of the ankle, knee, and hip, respectively. ${}^A_B\vec{X} = [x\ z\ \alpha]^T$ is the coordinates of frame {B} measured with respect to frame {A}, where $[x\ z]^T$ are the coordinates of the origin of frame {B} and α is the angle as indicated in Figure A-1. The lower link (shank) is of length l_1, while the upper link (thigh) is of length l_2. The foot is assumed to be resting on level ground.

The forward kinematic mapping from $\vec{\Theta} = [\theta_h\ \theta_k\ \theta_a]^T$ to ${}^A_B\vec{X} = [x\ z\ \alpha]^T$ is as follows:

$$ {}^A_B\vec{X} = \begin{bmatrix} x \\ z \\ \alpha \end{bmatrix} = \begin{bmatrix} -l_1\ \sin(\theta_a) - l_2\ \sin(\theta_a + \theta_k) \\ l_1\ \cos(\theta_a) + l_2\ \cos(\theta_a + \theta_k) \\ -\theta_h - \theta_k - \theta_a \end{bmatrix} = \vec{G}(\vec{\Theta}) \tag{A.1} $$

The Jacobian ${}^A_BJ = \frac{\partial \vec{G}}{\partial \vec{\Theta}}$ is obtained as follows:

$$ {}^A_BJ = \begin{bmatrix} -l_1\ \cos(\theta_a) - l_2\ \cos(\theta_a + \theta_k) & -l_2\ \cos(\theta_a + \theta_k) & 0 \\ -l_1\ \sin(\theta_a) - l_2\ \sin(\theta_a + \theta_k) & -l_2\ \sin(\theta_a + \theta_k) & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{A.2} $$

The Jacobian can be viewed as a mapping from the velocity in $\vec{\Theta}$ to those in ${}^A_B\vec{X}$:

$$ {}^A_B\dot{\vec{X}} = {}^A_B J\, \dot{\vec{\Theta}} \tag{A.3} $$

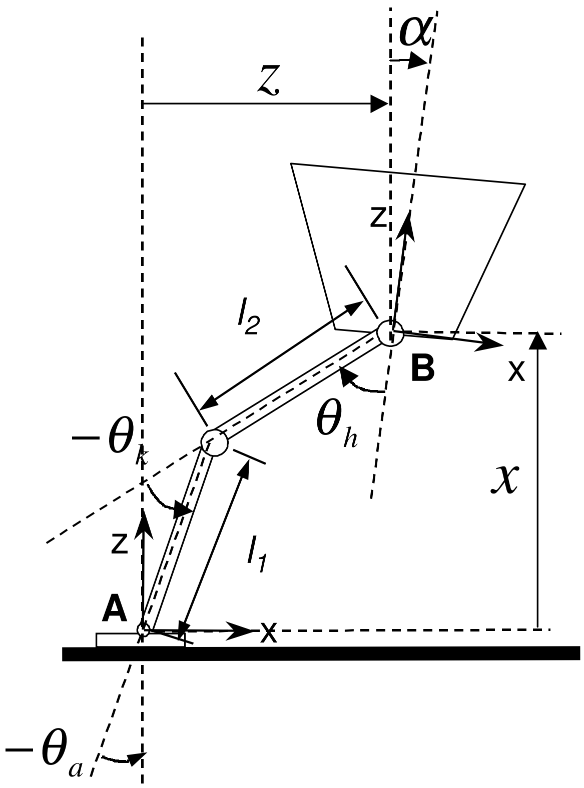


Figure A-1: A planar four-link, three-joint model which is used to represent the stance leg of the bipedal robot. Coordinate frame {B} is attached to the body and frame {A} is attached to the foot.

The Jacobian can also be used in the force domain by using the principle of *virtual work*. Let ${}^{A}_{B}\vec{F}$ be the virtual force-moment vector acting at the origin of the frame {B} (expressed in frame {A}); and $\vec{\tau}$ be the vector of torques associated to the joints coordinate $\vec{\Theta}$. $\vec{\tau}$ can be obtained from ${}^{A}_{B}\vec{F}$ using the Jacobian transpose [5]:

$$\vec{\tau} = ({}^{A}_{B}J)^{T}\ ({}^{A}_{B}\vec{F}) \tag{A.4}$$

With a limp ankle ($\tau_a = 0$), Equation A.4 is as follows:

$$\begin{bmatrix} 0 \\ \tau_k \\ \tau_h \end{bmatrix} = \begin{bmatrix} -l_1\ \cos(\theta_a) - l_2\ \cos(\theta_a + \theta_k) & -l_1\ \sin(\theta_a) - l_2\ \sin(\theta_a + \theta_k) & -1 \\ -l_2\ \cos(\theta_a + \theta_k) & -l_2\ \sin(\ \theta_a + \theta_k) & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} f_x \\ f_z \\ m_\alpha \end{bmatrix} \tag{A.5}$$

Due to the constraint at the ankle, the values of the components of the virtual force-moment vector $[f_x\ f_z\ m_\alpha]^T$ can no longer be chosen arbitrarily. In this thesis, it is desirable to control the vertical height and body pitch of the bipeds during the single support phase. Thus, the virtual force f_z and virtual moment m_α are arbitrary, whereas f_x can then be determined by the following equation:

$$f_x = \frac{-1}{l_1\ \cos(\theta_a) + l_2\ \cos(\theta_a + \theta_k)} \begin{bmatrix} l_1\ \sin(\theta_a) + l_2\ \sin(\theta_a + \theta_k) & 1 \end{bmatrix} \begin{bmatrix} f_z \\ m_\alpha \end{bmatrix} \tag{A.6}$$

By substituting Equation A.6 into Equation A.5, the following equation that gives the hip and knee joint torques given the virtual forces $[f_z\ m_\alpha]^T$:

$$\begin{bmatrix} \tau_k \\ \tau_h \end{bmatrix} = \begin{bmatrix} \frac{-l_1 l_2\ \sin(\theta_k)}{l_1\ \cos(\theta_a) + l_2\ \cos(\theta_a + \theta_k)} & \frac{-l_1\ \cos(\theta_a)}{l_1\ \cos(\theta_a) + l_2\ \cos(\theta_a + \theta_k)} \\ 0 & -1 \end{bmatrix} \begin{bmatrix} f_z \\ m_\alpha \end{bmatrix} \tag{A.7}$$

Note that the matrix in Equation A.7 is of full rank for all values of $\vec{\Theta}$ except for $\theta_k = 0$. This corresponds to a straightened knee, during which the admissible displacement components in ${}^{A}_{B}\vec{X}$ are no longer independent. In the force domain, it means that there is a many-to-one mapping from the virtual force vector $[f_z\ m_\alpha]^T$ to the torque vector $[\tau_k\ \tau_h]^T$. Equation A.7 is used in Chapter 5 for the stance leg control of the bipedal robot in the sagittal plane to achieve height and body pitch regulation.

Appendix B

Image Sequences from Videos

This appendix includes selected digitized image sequences of the simulation results. The image sequences in Figures B-1 and B-2 are extracted from the 3D bird-like walking implementation of the simulated M2. The algorithm can be found in Section 6.4. The image sequence in Figure B-3 is from the 3D human-like walking implementation whose algorithm can be found in Section 7.2.

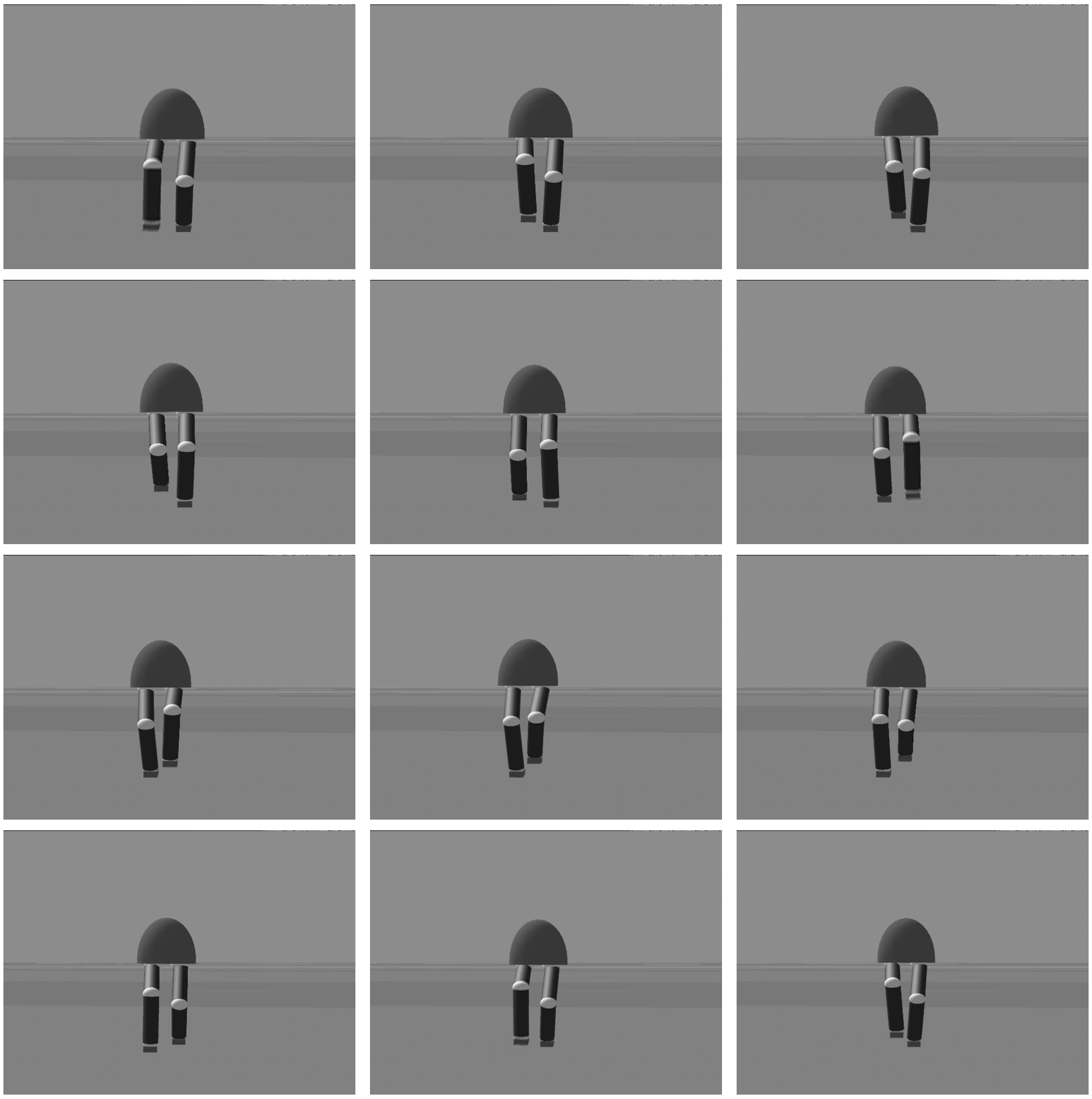


Figure B-1: The rear view image sequence (left to right, top to bottom) of the 3D bird-like walking implementation of the simulated M2. The control algorithm and implementation parameters can be found in Section 6.4. The images are $0.13s$ apart.

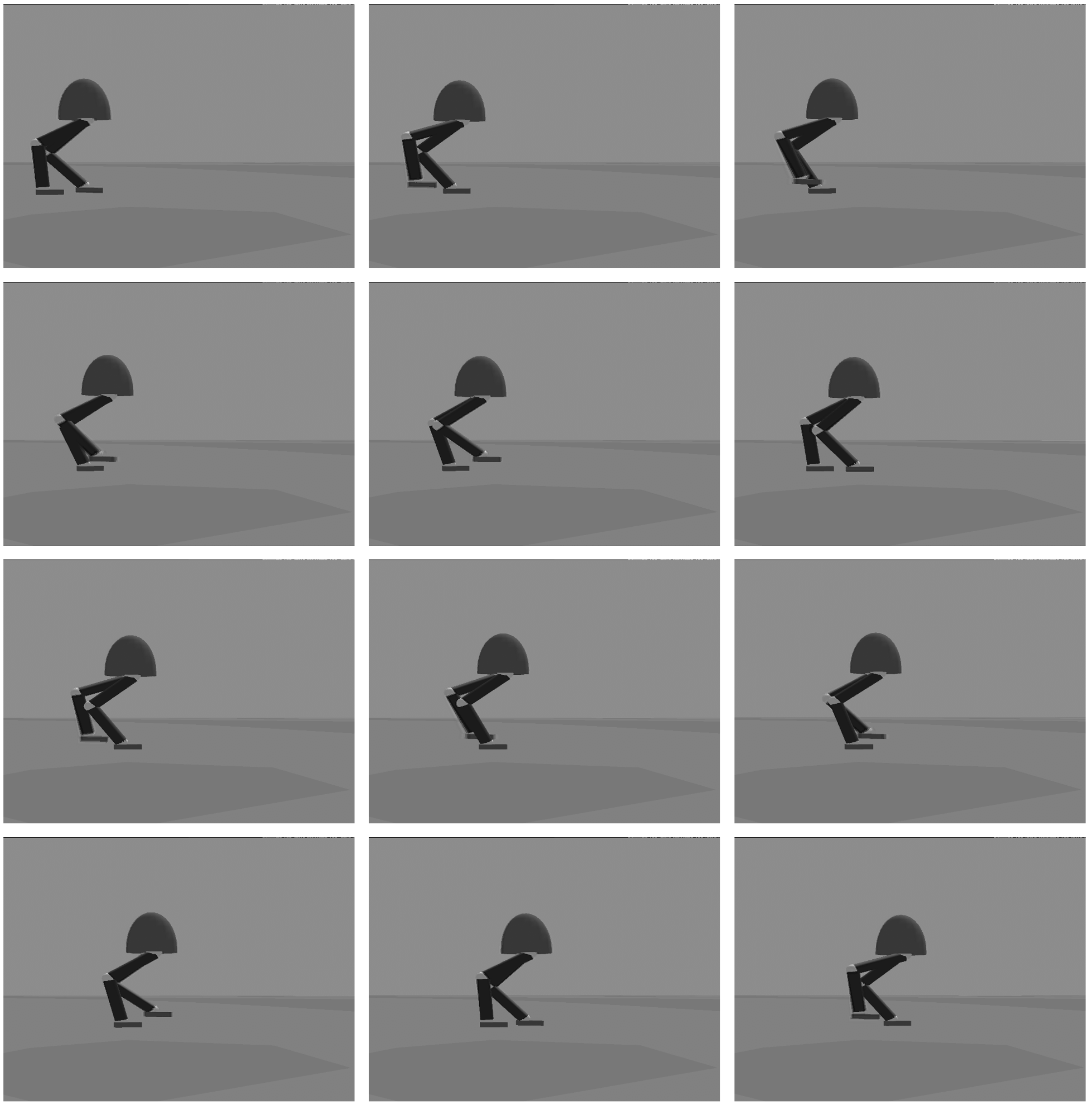


Figure B-2: The side view image sequence (left to right, top to bottom) of the 3D bird-like walking implementation of the simulated M2. The control algorithm and implementation parameters can be found in Section 6.4. The images are 0.13s apart.

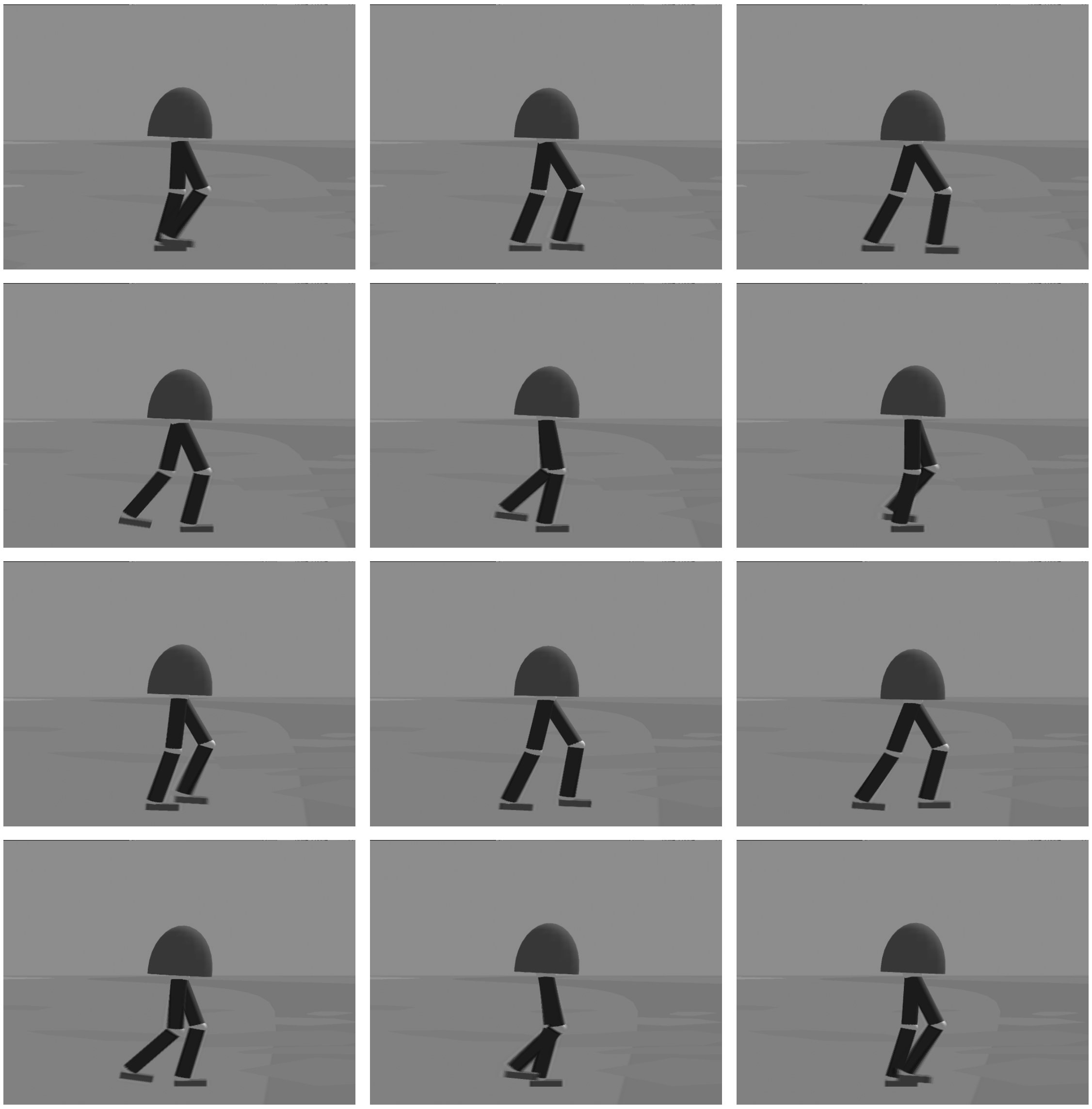


Figure B-3: The side view image sequence (left to right, top to bottom) of the 3D human-like walking implementation of the simulated M2. The control algorithm and implementation parameters can be found in Section 7.2. The images are $0.1s$ apart.

Bibliography

[1] J. Adolfsson, H. Dankowicz, and A. Nordmark. 3-d stable gait in passive bipedal mechanisms. *Proceedings of 357 Euromech*, 1998.

[2] J. S. Albus. Data storage in the cerebellar model articulation controller (cmac). *Journal of Dynamic Systems, Measurement, and Control*, pages 228–233, Sept. 1975.

[3] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller. *Journal of Dynamic Systems, Measurement, and Control*, pages 220–227, Sept. 1975.

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

[5] H. Asada and J.-J. E. Slotine. *Robot Analysis and Control*. John Wiley and Sons, 1986.

[6] A. G. Barto. Reinforcement learning. In O. Omidvar and D. L. Elliott, editors, *Neural Systems for Control*, chapter 2, pages 7–30. Academic Press, 1997.

[7] J. S. Bay and H. Hemami. Modeling of a neural pattern generator with coupled nonlinear oscillators. *IEEE Transactions on Biomedical Engineering*, BME-34(4):297–306, April 1987.

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

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

[10] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[11] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Volume One*. Athena Scientific, Belmont, MA, 1995.

[12] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Volume Two*. Athena Scientific, Belmont, MA, 1995.

[13] J. A. Boyan and A. W. Moore. Generatization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. K. Keen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge, MA, 1995.

[14] R. Brooks and M. Mataric. Real robots, real learning problems. In J. Connell and S Mahadevan, editors, *Robot Learning*. Kluwer Academic Publishers, 1993.

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

[16] G. Burgin. Using cerebellar arithmetic computers. *AI Expert*, pages 32–41, 1992.

[17] C.-M. Chew and G. Pratt. A general control architecture for dynamic bipedal walking. In *IEEE International Conference on Robotics and Automation*, 2000.

[18] Chee-Meng Chew. Blind walking of a planar biped over sloped terrain. Master's thesis, Massachusetts Institute of Technology, Feb 1998.

[19] Chee-Meng Chew and Gill A. Pratt. A minimum model adaptive control approach for a planar biped. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyongju, Korea*, 1999.

[20] 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.

[21] M.J. Coleman and A. Ruina. An uncontrolled walking toy that cannot stand still. *Physical Review Letters*, pages 3658–3661, April 1998.

[22] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.

[23] R. Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, (210):492–498, 1980.

[24] R. Q. Van der Linde. Active leg compliance for passive walking. *IEEE International Conference on Robotics and Automation*, pages 2339–2345, 1998.

[25] Eric Dunn and Robert Howe. Towards smooth bipedal walking. *IEEE International Conference on Robotics and Automation*, pages 2489–2494, 1994.

[26] Eric Dunn and Robert Howe. Foot placement and velocity control in smooth bipedal walking. *IEEE International Conference on Robotics and Automation*, pages 578–583, 1996.

[27] T. Flash and N. Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience*, 5(7):1688–1703, 1985.

[28] J. Fowble and A. Kuo. Stability and control of passive locomotion in 3d. *Proceedings of the Conference on Biomechanics and Neural Control of Movement*, pages 28–29, 1996.

[29] L. Fu. *Neural Networks in Computer Intelligence*. McGraw-Hill, 1994.

[30] M. Garcia, A. Chatterjee, and A. Ruina. Speed, efficiency, and stability of small-slope 2d passive dynamic bipedal walking. *IEEE International Conference on Robotics and Automation*, pages 2351–2356, 1998.

[31] J. A. Golden and Y. F. Zheng. Gait synthesis for the sd-2 biped robot to climb stairs. *International Journal of Robotics and Automation*, 5(4):149–159, 1990.

[32] C. L. Golliday and H. Hemami. An approach to analyzing biped locomotion dynamics and designing robot locomotion controls. *IEEE Transactions on Automatic Control*, AC-42:963–973, Dec. 1977.

[33] A. Goswami, B. Espiau, and A. Keramane. Limit cycles in a passive compass gait biped and passivity-mimicking control laws. *Journal of Autonomous Robots*, 1997.

[34] S. Grillner. Locomotion in vertebrates: Central mechanisms and reflex interaction. *Physiological Reviews*, (55):247–304, 1975.

[35] A. A. Grishin, A. M. Formal'sky, A. V. Lensky, and S. V. Zhitomirsky. Dynamic walking of a vehicle with two telescopic legs controlled by two drives. *International Journal of Robotics Research*, 13(2):137–147, 1994.

[36] F. Gubina and H. Hemami R. B. McGhee. On the dynamic stability of biped locomotion. *IEEE Transactions on Biomedical Engineering*, BME-21(2):102–108, Mar 1974.

[37] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Upper Saddle River, New Jersey, 1999.

[38] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. *IEEE International Conference on Robotics and Automation*, 1998.

[39] Y. Hurmuzlu. Dynamics of bipedal gait: Part i - objective functions and the contact event of a planar five-link biped. *Journal of Applied Mechanics*, 60:331–336, 1993.

[40] Y. Hurmuzlu. Dynamics of bipedal gait: Part ii - stability analysis of a planar five-link biped. *Journal of Applied Mechanics*, 60:337–343, 1993.

[41] V. T. Inman, H. J. Ralson, and F. Todd. *Human Walking*. Willaims and Wilkins, Baltimore, London, 1981.

[42] T. Jaakkola, S. P. Singh, and M. I. Jordan. On the convergence of stochastic iterative dynamics programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[43] L. Jalics, H. Hemami, and B. Clymer. A control strategy for adaptive bipedal locomotion. *IEEE International Conference on Robotics and Automation*, pages 563–569, 1996.

[44] Fred R. Sias Jr. and Yuan F. Zheng. How many degrees-of-freedom does a biped need? *IEEE International Conference on Intelligent Robots and Systems*, pages 297–302, 1990.

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

[46] S. Kajita and K.Tani. Experimental study of biped dynamic walking in the linear inverted pendulum mode. *IEEE International Conference on Robotics and Automation*, pages 2885–2891, 1995.

[47] S. Kajita and K.Tani. Adaptive gait control of a biped robot based on realtime sensing of the ground profile. *IEEE International Conference on Robotics and Automation*, pages 570–577, 1996.

[48] Shuuji Kajita and Kazuo Tani. Study of dynamic biped locomotion on rugged terrain - derivation and application of the linear inverted pendulum mode. *IEEE International Conference on Robotics and Automation*, pages 1405–1412, 1991.

[49] Shuuji Kajita and Kazuo Tani. Study of dynamic biped locomotion on rugged terrain - theory and basic experiment. *Fifth International Conference on Advanced Robotics (ICAR'91)*, pages 741–746, 1991.

[50] Shuuji Kajita and Kazuo Tani. Experimental study of biped dynamic walking. *IEEE Control Systems Magazine*, pages Feb:13–19, 1996.

[51] Shuuji Kajita, Kazuo Tani, and Akira Kobayashi. Dynamic walk control of a biped robot along the potential energy conserving orbit. *IEEE International Conference on Intelligent Robots and Systems*, pages 789–794, 1990.

[52] Shuuji Kajita, Tomio Yamaura, and Akira Kobayashi. Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Transactions on Robotics and Automation*, 6(1):431–438, 1992.

[53] I. Kato and H. Tsuiki. Hydraulically powered biped walking machine with a high carrying capacity. *Proceedings of the 4th International Symposium on External Control of Human Extremities*, pages 410–421, 1972.

[54] T. Kato, A. Takanishi, H. Ishikawa, and I. Kato. The realization of the quasi dynamic walking by the biped walking machine. *Romansy '81*, pages 341–351, 1981.

[55] R. Katoh and M. Mori. Control method of biped locomotion giving asymptotic stability of trajectory. *Automatica*, 20(4):405–414, 1984.

[56] Hassan K. Khalil. *Nonlinear Systems.* Prentice-Hall, Upper Saddle River, NJ, 1996.

[57] Andrew L. Kun and Thomas W. Miller III. Adaptive dynamic balance of a biped robot using neural networks. *IEEE International Conference on Robotics and Automation*, pages 240–245, 1996.

[58] Andrew L. Kun and Thomas W. Miller III. Unified walking control for a biped robot using neural networks. *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, pages 283–288, 1998.

[59] Andrew L. Kun and Thomas W. Miller III. Control of variable speed gaits for a biped robot. *IEEE Robotics and Automation Magazine*, (63):19–29, 1999.

[60] Paul W. Latham. *A Simulation Study of Bipedal Walking Robots: Modeling, Walking Algorithms, and Neural Network Control*. PhD dissertation, University of New Hampshire, Sept 1992.

[61] Cynthia R. Lee and Claire T. Farley. Determinants of the center of mass trajectory in human walking and running. *Journal of Experimental Biology*, 201:2935–2944, 1998.

[62] Chun-Shin Lin and Hyongsuk Kim. Cmac-based adaptive critic self-learning control. *IEEE Transactions on Neural Networks*, 2(5):530–533, 1991.

[63] Tad McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9(2):62–82, 1990.

[64] Tad McGeer. Passive walking with knees. *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1640–1645, 1990.

[65] Thomas A. McMahon. *Muscles, Reflexes, and Locomotion*. Princeton University Press, Princeton, New Jersey, 1984.

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

[67] Thomas W. Miller and Filson H. Glanz. The university of new hampshire implementation of the cerebellar model arithmetic computer - cmac. Unpublished reference guide to a CMAC program, August 1994.

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

[69] W. T. Miller, P. J. Latham, and S. M. Scalera. Bipedal gait adaptation for walking with dynamic balance. In *Proceedings of American Control*, pages 1603–1608, 1991.

[70] T. Mita, T. Yamaguchi, T. Kashiwase, and T. Kawase. Realization of a high speed biped using modern control theory. *International Journal of Control*, 40(1):107–119, 1984.

[71] H. Miura and I. Shimoyama. Dynamic walk of a biped. *International Journal of Robotics Research*, 3(2):60–74, 1984.

[72] Simon Mochon and Thomas A. McMahon. Ballistic walking: An improved model. *Mathematical Biosciences*, 52:241–260, 1979.

[73] Richard M. Murray and John Hauser. A case study in approximate linearization: The acrobat example. Technical Report UCB/ERL (unknown), University of California, Berkeley, College of Engineering, Berkeley, California, 1991.

[74] Eadweard Muybridge. *The Human Figure in Motion*. Dover Publications, New York, NY, 1955.

[75] Eadweard Muybridge. *Animals in Motion*. Dover Publications, New York, NY, 1957.

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

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

[78] Gill A. Pratt. Legged robots at mit - what's new since raibert. *2nd International Conference on Climbing and Walking Robots*, pages 29–33, 1999.

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

[80] 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.

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

[82] Jerry E. Pratt and Gill A. Pratt. Exploiting natural dynamics in the control of a planar bipedal walking robot. *Proceedings of the Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing*, pages 739–748, 1998.

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

[84] 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.

[85] M. H. Raibert and F. C. Wimberly. Tabular control of balance in a dynamic legged system. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(2):324–339, 1984.

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

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

[88] R. Ringrose. The creature library. Unpublished reference guide to a C library used to create physically realistic simulations, 1992.

[89] David W. Robinson, Jerry E. Pratt, Daniel J. Paluska, and Gill A. Pratt. Series elastic actuator development for a biomimetic robot. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 1999.

[90] Jessica Rose and James G. Gamble. *Human Walking*. Williams and Wilkins, 1994.

[91] D. E. Rosenthal and M. A. Sherman. High performance multibody simulations via symbolic equation manipulation and kane's method. *Journal of Astronautical Sciences*, 34(3):223–239, 1986.

[92] M. Rostami and G. Bessonnet. Impactless sagittal gait of a biped robot during the single support phase. *IEEE International Conference on Robotics and Automation*, pages 1385–1391, 1998.

[93] G. A. Rummery. *Problem Solving with Reinforcement Learning*. PhD dissertation, Cambridge University, UK, 1995.

[94] A. W. Salatian and Y.-F. Zheng. Gait synthesis for a biped robot climbing sloping surfaces using neural networks - part ii: Dynamic learning. *IEEE International Conference on Robotics and Automation*, pages 2601–2611, 1992.

[95] A. W. Salatian and Yuan-F. Zheng. Gait synthesis for a biped robot climbing sloping surfaces using neural networks - part i: Static learning. *IEEE International Conference on Robotics and Automation*, pages 2601–2611, 1992.

[96] C.-L. Shih and C. J. Chiou. The motion control of a statically stable biped robot on an uneven floor. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(2):244–249, 1998.

[97] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice-Hall, Cambridge, MA, 1991.

[98] M. W. Spong. Swing up control of the acrobot. In *Proceedings of the 1994 IEEE Conference on Robotics and Automation*, pages 2356–2361, 1994.

[99] Steve Stitt and Yuan-F. Zheng. Distal learning applied to biped robots. *IEEE International Conference on Robotics and Automation*, pages 137–142, 1994.

[100] S. Strogatz. *Nonlinear Dynamics and Chaos*. Addison-Wesley, Reading, MA, 1994.

[101] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Systems 8*, pages 1038–1044, 1996.

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

[103] Gentaro Taga. A model of the neuro-musculo-skeletal system for human locomotion: I. emergence of basic gait. *Biological Cybernetics*, 73:97–111, 1995.

[104] Gentaro Taga. A model of the neuro-musculo-skeletal system for human locomotion: Ii. real-time adaptability under various constraints. *Biological Cybernetics*, 73:113–121, 1995.

[105] A. Takanishi, M. Ishida, Y. Yamazaki, and I. Kato. The realization of dynamic walking by the biped walking robot wl-10rd. *Proceedings of ICAR '85*, pages 795–801, 1985.

[106] Atsuo Takanishi, Hun-Ok Lim, Masayuki Tsuda, and Ichiro Kato. Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface. *IEEE International Workshop on Intelligent Robots and Systems*, pages 323–330, 1990.

[107] Atsuo Takanishi, Tomoaki Takeya, Hideyuki Karaki, and Ichiro Kato. A control method for dynamic biped walking under unknown external force. *IEEE International Workshop on Intelligent Robots and Systems*, pages 795–801, 1990.

[108] Kazuo Tani, Kiichi Ikeda, Tomoaki Yano, Shuuji Kajita, and Osarnu Matsumoto. The concept of model free robotics for robots to act in uncertain environments. *Proceedings of the 1993 IEEE/Tsukuba International Workshop on Advanced Robotics*, pages 85–90, 1993.

[109] E. Thelen and L. B. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action.* MIT Press, 1994.

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

[111] John N. Tsitsiklis. Asynchoronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202, 1994.

[112] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic. *Biped Locomotion: Dynamics, Stability, Control, and Applications.* Springer-Verlag, Berlin, 1990.

[113] H. Wang, T. T. Lee, and W. A. Grover. A neuromorphic controller for a three-link biped robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(1):164–169, Jan/Feb 1992.

[114] C. J. C. H. Watkins. *Learning from Delayed rewards.* PhD dissertation, Cambridge University, UK, 1989.

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

[116] B. Widrow and M. Hoff. Adaptive swing circuits. In *1960 IRE WESCON Convention Record*, pages 96–104, 1960.

[117] Matthew M. Williamson. Series elastic actuators. Master's thesis, Massachusetts Institute of Technology, June 1995.

[118] Mark W.Spong. The swing up control problem for the acrobot. *IEEE Control Magazine*, pages 49–55, Feb 1995.

[119] Jin'ichi Yamaguchi, Noboru Kinoshita, Atsuo Takanishi, and Ichiro Kato. Development of a dynamic biped walking system for humanoid - development of a biped walking robot adapting to the humans' living floor. *IEEE International Conference on Robotics and Automation*, pages 232–239, 1996.

[120] Jin'ichi Yamaguchi, Daisuke Nishino, and Atsuo Takanishi. Realization of dynamic biped walking varying joint stiffness using antagonistic driven joints. *IEEE International Conference on Robotics and Automation*, pages 2022–2029, 1998.

[121] Jin'ichi Yamaguchi, Eiji Soga, Sadatoshi Inoue, and Atsuo Takanishi. Development of a bipedal humanoid robot - control method of whole body cooperative dynamic biped walking. *IEEE International Conference on Robotics and Automation*, pages 368–374, 1999.

[122] Jin'ichi Yamaguchi, Atsuo Takanishi, and Ichiro Kato. Development of a biped walking robot compensating for three-axis moment by trunk motion. *IEEE International Conference on Intelligent Robots and Systems*, pages 561–566, 1993.

[123] Jin'ichi Yamaguchi, Atsuo Takanishi, and Ichiro Kato. Development of a biped walking robot adapting to a horizontally uneven surface. *IEEE International Conference on Intelligent Robots and Systems*, pages 1156–1163, 1994.

[124] Keon-Young Yi and Yuan-F. Zheng. Biped locomotion by reduced ankle power. *IEEE International Conference on Robotics and Automation*, pages 584–589, 1996.

[125] Yuan F. Zheng and Jie Shen. Gait synthesis for the sd-2 biped robot to climb sloping surfaces. *IEEE Transactions on Robotics and Automation*, 6(1):86–96, 1990.