Imitation-based Learning of Bipedal Walking Using Locally Weighted Learning

by

Kevin Loken

B.A.Sc., Simon Fraser University, 1992

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

August 2006

Abstract

Walking is an extremely challenging problem due to its dynamically unstable nature. It is further complicated by the high dimensional continuous state and action spaces. We use locally weighted projection regression (LWPR) as a locally structurally adaptive nonlinear function approximator as the basis for learned control policies. Empirical evidence suggests that control policies for high dimensional problems exist on low dimensional manifolds. The LWPR algorithm models this manifold in a computationally efficient manner as it only models those states which are visited using a local dimensionality reduction technique based on partial least squares regression.

We show that local models are capable of learning control policies for physics-based simulations of planar bipedal walking. Locally structured control policies are learned from observation of a variety of different inputs including observation of human control and existing parametrized control policies. We extend the pose control graph to the concept of policy control graph and show that this representation allows for the learning of transition points between different control policies.

Kevin Loken

University of British Columbia
August 2006

Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgements

I wish to thank my supervisor, Michiel van de Panne, for his continuous enthusiasm, his interest in so many areas of computer science, and his willingness to let me go off on tangents for weeks on end. I also wish to thank my second reader Ian Mitchell for his thorough review of this thesis. The clarity of many of the explanations is due in large part to his comments.

No acknowledgment section is complete with the requisite thanks to previous bullpen mates and current lab mates. Mike Yurick, David White, Tyson Brochu, Kang Kang Yin, and Phillipe Beaudoin made it a great pleasure to venture on to campus each week.

Special thanks go to my parents for instilling in me a love of learning from a very young age. Last, but definitely not least, I have to thank my family — my wife Susan, children Sarah, David, Andrew, and Graham — for allowing me the opportunity to leave my job and come back to school.

Chapter 1

Introduction

Humanoid robots have long been a fascination of man-kind. First visualized in the 1926 silent movie *Metropolis* [23, 31] the vision of intelligent and dynamic humanoid robots has been a powerful symbol. Yet, despite nearly eighty years of these images we are only now beginning to see some progress in mobile humanoid robots. Engineering marvels like the Honda Asimo [49] are impressive in their accomplishments, able to travel at a steady walk, run at up to 6 km/h, walk up and down stairs, and perform simple tasks such as pushing a cart or carrying a tray of coffee mugs. The Honda team has even made Asimo perform traditional Japanese folk dances [33]. However, these impressive machines are still sorely lacking in truly dynamic maneuvers such as those involved in playing any ordinary game of soccer. These limitations are imposed both by the limits of the electromechanical design of the system and the inherent complexity of the control algorithms which must be developed.

1.1 Imitating skillful human motion

The imitation of skillful human motion is useful both from a computer animation stand point and in the realm of robotics. For computer animation, the ability to create physically realistic motion quickly and easily is desirable both in the film industry and in the interactive entertainment (video game) industry. Currently animation is either generated by hand through the talents of skilled animators

or it is replicated from human motion capture data.

The former method is time consuming and expensive, and if certain aspects of a motion change then whole new motion segments need to be created. The use of motion capture aids in this regard, but it comes with its own issues: the actor used to generate the motion is not necessarily of the same size as the avatar that will display the motion. How does an ogre move anyway? Having avatars that are capable of performing realistic-looking, and physically plausible, motion would greatly improve the quality of the animation in interactive entertainment.

In robotics, projects such as the NASA Robonaut [2] aim to produce humanoid robots that are capable of performing the same actions as the astronauts they would replace. This raises the question of how you create a control system that can replicate the many different tasks the Robonaut would have to perform, from performing a space-walk to using a screwdriver.

The study of how humans plan and execute skilled motion is a vast area of research that is extraordinarily interdisciplinary in its nature. Researchers from such diverse fields as anatomy, control theory, robotics, machine learning, bio-mechanics, kinesiology and neuroscience all study the problem. Each field approaches the problem with its own set of techniques, motivations and constraints.

1.2 Why is motor control hard?

On the face of it, one would think that motor control is not a particularly difficult problem to solve. After all, we all learn to perform hundreds, possibly even thousands, of skilled actions in our lifetime. We learn to walk at around one year of age purely from observation and trial and error. We walk without thinking about it, and are capable of navigating varying terrain with ease. This apparent ease with which we all perform these actions belies the underlying

complexity of the problem.

The goals of each field change the underlying assumptions that researchers make, and the types of approximations that they bring to a problem. While a computer scientist might be interested in applying dynamics to a generated computer animation to make it more realistic looking, a bio-mechanics researcher might be interested in exactly computing forces and torques involved in a motion for the design of a new prosthetic limb. These two different goals will yield different techniques for attempting to solve a particular dynamics problem, one concerned with speed of computation, the other with accuracy of results.

Modeling Issues

There are many different ways in which a human can be represented for a simulation. Common representations use idealized joints and idealized motors to power motions. For example, the knee joint is usually modeled as a single degree of freedom joint with an idealized motor that can move the calf. In reality, the joint consists of four bones coupled together through flexible tendons and muscles, cushioned by cartilage and powered through the contraction of hundreds (thousands) of antagonistic and synergistic muscle fibers.

Control Issues

A human is also an extremely high dimensional mechanism. When performing an action such as walking, hundreds of muscle fibers are activating in a coordinated fashion to drive the overall motion while maintaining balance. There are easily more than one hundred controllable degrees of freedom embedded in an equally as large state space. Even idealized models of humans such as video game characters may have as many as thirty controllable degrees of freedom embedded in a sixty dimensional state space. This brings up Bellman's *curse of*

dimensionality [5] which states that complexity of control grows *exponentially* in the number of dimensions.

With a motion such as walking we are also faced with a credit assignment problem. When a robot falls over it is usually not a result of the action just taken, but as the result of some action taken in the past. When a control strategy fails identifying the "wrong" action is nearly impossible. Learning algorithms must often employ techniques such as eligibility traces [46] to update their internal state. This ultimately slows the learning process requiring many more trials to reach a stable control strategy.

Simulation Issues

Once a researcher has chosen their approach and built their computing infrastructure, they can then go about simulating the physics of human motion. Unfortunately, the problems do not end here. Depending on the physical representation used and the detail used in the model (e.g. motors vs. muscles, inclusion of tendon dynamics) simulation times may be long. It is also possible that the simulation will not produce the desired result. Sources of failure can be due to defects in the coding of the algorithm, numerical instability, or the inherent instability of the model under control. With a simulation of dynamically unstable action such as walking, if there is even a single error in one component the robot is likely to lose balance or fail in its task in some other way.

1.3 Recent Progress

Given that trying to teach a robot how to perform a skilled motion like walking seems so difficult, why is now such a good time to be involved in this area of research? The interdisciplinary nature of motor control research is both a curse and a blessing. The curse is that in order to make progress on the problem we

must gain an understanding of several different areas of research. The blessing is that researchers from multiple fields are tackling the problem from different points of view. Promisingly we are beginning to see convergence in the various computational, mechanical, neuro- and biological aspects of motor control.

Over the last fifteen years there has been significant advancement in the control of walking robots. In two seminal papers Tad McGeer showed that a purely passive mechanical system is capable of walking, provided it is given enough energy to overcome friction [26, 27]. This spawned an area of research known as passive dynamic walking that aims to leverage the natural mechanics of leg configurations to produce highly efficient walk cycles. Trajectory tracking approaches from robotics, modulated with zero-moment point (ZMP) balance control, have been successfully applied to create walking robots such as Honda's Asimo [47, 49].

Moore's Law is a blessing and in the last decade computing power has increased substantially so what were once intractable simulation problems are now routinely solved in a few seconds. This allows numerical optimization techniques such as space-time constraints [58] to be used to generate physically valid joint torque trajectories.

Machine learning techniques have also improved substantially in the last decade, and various imitation based learning algorithms exist that can generalize from relatively few points of observed data.

Lastly, the rise of video games and computer generated animated films has pushed the advancement of techniques for recording human motion. Mechanical, magnetic and optical methods have been perfected for observing human motion and generating joint angle trajectories from the recorded motion. These motion capture systems have allowed researchers to capture many types of human motion and millions of frames of animation quickly and easily. This provides a

vast repository of human motion data that can be used as reference solutions when developing control techniques.

1.4 Goals

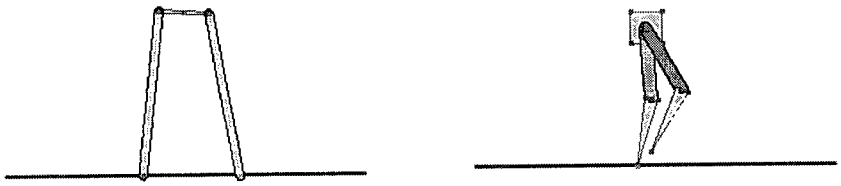
The grand vision of this thesis is to find a method for providing robots with the ability to learn the way humans do, through observation and trial and error. The learned information should then be placed within a framework that lets controllers be shared and that thus endows robots with multiple skills that can be sequenced in time.

To reduce the scope of this problem, this thesis tries to answer the following more specific questions:

- Can control strategies involving balance, such as walking, be learned from observation of a known solution? This observation can take at least two forms, either purely kinematic observations like motion capture data or through known control actions.
- Can locally weighted learning techniques such as locally weighted projection regression (cf. Chapter 4) be applied effectively for difficult control tasks involving under-actuated systems?
- Can the locally weighted learning techniques be used to support transitions between different classes of motion, such as walking at different speeds?

The previous work related to these questions will be discussed in Chapter 2 and Chapter 4.

Balance is a challenging problem, as the control strategy needs to incorporate the numerous small corrective actions that are layered on top of the large scale gross motor movement of walking. It is unclear whether a learning algorithm



(a) A 3-link planar biped. (b) A 5-link planar biped with knees.

Figure 1.1: The two planar biped models used in the experiments

will be able to identify these corrective behaviors given the various sources of noise within the observations of motion. The various locally-weighted learning algorithms that exist seem promising in their ability to control certain types of robotic systems, though to date they seem to have been applied to fully actuated systems.

Most current work in motor control also only addresses skills in isolation such as walking at a fixed speed. Walks of different speeds provide a well defined family of motions for testing the ability of learned locally defined control policies to cope with transition motions.

1.5 Overview of Approach

Models

Throughout the experiments presented in Chapter 5 the two models shown in Figure 1.1 were used. These are planar biped models, restricted to two dimensional motion. This use of planar bipeds allows for much faster simulation times, with the focus being on the learning of control policies from observations.

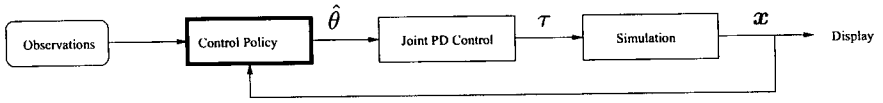


Figure 1.2: A simple block diagram of the simulation. The bold block denoting the control policy is the main focus of this thesis. We wish to use observation data to learn the control policy π.

Control Structure

The overall simulation structure is shown in Figure 1.2. Observations of a control policy are used in a supervised learning algorithm (cf. Chapter 4) to create an approximate control policy $\hat{\pi}$. The control policy outputs target joint angles, $\hat{\theta}$, based on the current system state $\boldsymbol{x}$ or a projection of the current system state. These target joint angles are passed to a joint proportional derivative controller which converts them to joint torques τ. The joint torques are used by the simulation to compute accelerations, including forces generated by interaction with the ground (cf. Chapter 3). The resulting accelerations are integrated with a simple forward Euler scheme.

The details of each block are presented in the associated chapters. The primary focus of this thesis is on the mechanism of learning the control policy $\hat{\pi}$ and the details of the other blocks could be replaced with equivalent systems, as there are many different choices and compromises within each of the implementations.

Locally Weighted Control Policies

We are attempting to learn a control policy of the form

$$\hat{\theta} = \pi(\boldsymbol{x}) \tag{1.1}$$

The observation data is assumed to contain examples of $\pi(\boldsymbol{x})$, thus we have

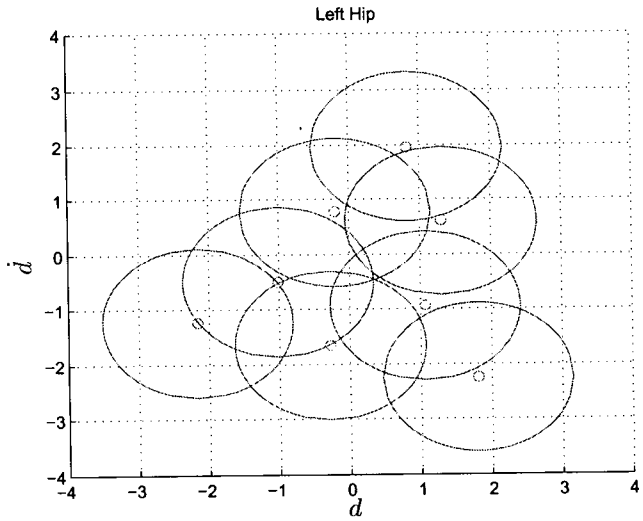


Figure 1.3: An example of how local linear models are created in the projected state space of a 3-link planar biped. The large ellipses represent the 0.1 weight contour of a Gaussian kernel function (Equation 4.2), and the small circles represent the centers of the receptive fields.

a supervised learning problem. The control law $\pi(\boldsymbol{x})$ is extracted through observation of joint torque trajectories during a walk cycle. Direct access to these joint torques is available in our simulations, though inverse dynamic techniques can be used to calculate them if a purely kinematic description of motion is available, such as motion capture data. The locally weighted projection regression algorithm builds locally linear models that approximate the non-linear function in Equation 1.1.

During the training of the locally weighted models, receptive fields are created that define the area of support for the locally linear approximations. An example of how these fields are created is shown in Figure 1.3. The contours are for a 10% weight, as defined by Equation 4.2. When receptive fields overlap, a weighted average of the linear approximations is used as the result, see Equation 4.3.

1.6 Contributions

The main contribution of this thesis is to show that locally weighted control policies are capable of controlling the walking gait of biped robots. The learned control policies are trained with observations of both human control and existing hard-coded control policies. The use of these techniques in control of walking has been limited to active walk gaits, with starting and stopping being handled through specialized controllers. Our application is capable of learning to initiate a walk cycle in addition to controlling the periodic gait.

We also extend the pose control graph [52] to that of the policy control graph. We show that this representation allows the development of transition points between different speeds of walk cycles allowing for planning of motion. This is shown through control of a planar biped simulation walking back and forth between two points.

1.7 Thesis Organization

The target audience of this thesis is a computer scientist experienced in kinematic animation techniques who is interested in applying this knowledge and expertise towards dynamic motion. Since motor control is such a difficult problem, and draws from so many different subject areas, the early parts of this thesis provide overviews of key areas that should provide sufficient background to understand the experiments performed in this thesis.

The rest of this thesis is organized as follows. Chapter 2 begins with definitions of many of the terms and concepts used in dynamics, motor control and machine learning. This is followed by an overview of related research which spans a large number of areas. Much of the related work is necessarily presented in terms of a high-level review with references representative of approaches,

rather than as an exhaustive list, with a few exceptions. An overview of the dynamic simulation used in this thesis is provided in Chapter 3. The specific machine learning algorithm, locally weighted project regression, is discussed in Chapter 4. Experimental results are presented in Chapter 5 and conclusions are presented in Chapter 6.

Chapter 2

Definitions and Related Work

The study of human walking covers a broad collection of disciplines, including bio-mechanics, robotics, kinematics, dynamics, signal processing and machine learning. We are concerned with the application and extension of existing techniques drawn from these many disciplines.

This chapter provides sufficient definitions and background material for an overall understanding and context of the work that was performed for this thesis. Readers who are interested in gaining a greater understanding of how individual components work should consult the bibliography and the various papers referenced throughout this section. There are also numerous conferences (CLAWAR, IROS, ICRA) that present current research on this and many related topics.

Additional information regarding the machine learning techniques used in this thesis are presented in Chapter 4.

2.1 Definitions

Because the study of human walking draws on so many different disciplines, there are often references to terms which may be unfamiliar to the reader. Many terms are also often overloaded with different meanings (e.g. "state"). Thus we provide specific explanations for the terms used throughout this thesis.

System State

The **state** of a system is defined as the smallest set of numbers that must be known in order that its future response to any given input can be calculated from the equations of motion [44] for a single point in time. This is a Markov model of the motion. This thesis uses articulated figures to represent idealized virtual humans. The hierarchical skeleton is a collection of local frames, each characterized by the position and orientation with respect to its parent frame. The set $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_N)$ of parameters corresponding to the degrees of freedom of the figure, together with their derivatives with respect to time $\dot{\boldsymbol{\theta}} = (\dot{\theta_1}, \dot{\theta_2}, \ldots, \dot{\theta_N})$ represent the **generalized coordinates** of the articulated figure. This generalized coordinate vector is referred to as the **full body state** in this thesis.

For our work, the state consist of all the relevant joint angles and velocities, as well as the global position and velocity, and global orientation and angular velocity of the root of the hierarchical skeleton, which for our representation is the hips. Each limb of the skeleton uses a thin-rod approximation of its inertia. The 3-link biped has the following state description:

$$\boldsymbol{x} = \left(x, \dot{x}, y, \dot{y}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, \theta_3, \dot{\theta}_3\right)$$

The state-space representation of a system is not unique, although it is minimum in terms of the number of dimensions and it is also the representation used by the dynamics simulation. There are many different sets of state variables that can be utilized for a given system.

For purposes of control, it is often useful to work with projections of the full state. One such useful projection is the center of mass position and velocity relative to the current stance foot. This is illustrated in Figure 2.1. This $(d, \dot{d})$

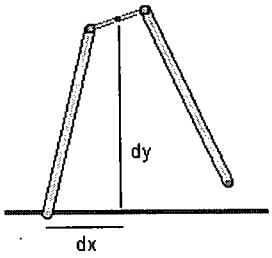


Figure 2.1: The projection of the full body state to the $(d, \dot{d})$ state for the 3-link biped. The center of mass state is $(d_x, \dot{d}_x, d_y, \dot{d}_y)$. In this thesis we only use $(d_x, \dot{d}_x)$ which we hereafter refer to as simply $(d, \dot{d})$.

parametrization provides sufficient information on the current phase of the walk cycle and allow control to be exerted on the biped. Note that the full body state can not be recovered from this $(d, \dot{d})$ projection as many different poses lead to the same projection.

Under-actuated systems

There is a large body of research that deals with the dynamics and control of robot structures. Much of this research is based on industrial style robots that are bolted to the ground, or world reference frame. By providing a joint between the root of the robot and the world reference frame, the robot is in principle able to achieve any desired set of joint accelerations.

In contrast humans, or bipedal robots, are **under-actuated** systems. The stance foot is not bolted to the ground and so not all possible accelerations are possible. By way of example, if you start pushing on a bipedal robot, the only way to maintain balance is to take a step. When the center of mass of the robot is sufficiently outside the base of support and leads to loss of balance and a fall. The robot is incapable of exerting a sufficient torque in any of its joints to

prevent this loss of balance, so the only option is to move the base of support (by taking a step). Recovery can also happen if the velocity of the center of mass is such that the momentum is sufficient to regain balance. Because these types of systems are under-actuated, control becomes much more difficult.

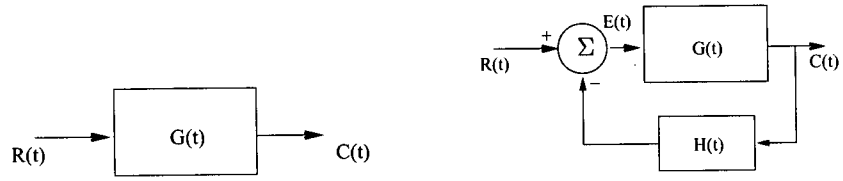
Open- and Closed-Loop Control

Control systems are often defined as **open loop** or **closed loop** depending on the feedback arrangement of the system. In control theory, the system under control is often referred to as the **plant**. In some motor control literature open loop control is often referred to as **feed-forward** control, while closed loop control is referred to as **feedback** control.

An open loop control system (Figure 2.2(a)) provides a time-varying reference signal signal $R(t)$ to the plant $G(t)$, regardless of the state of the plant. In a perfect world, with no errors in modeling and no perturbations created by the environment, this type of control is capable of producing the desired controlled output $C(t)$.

By contrast, a closed loop control system (Figure 2.2(b)) measures the current state of the system $C(t)$ transformed by an arbitrary feedback law $H(t)$ against the reference input $R(t)$ and applies an error correcting term $E(t)$. This improves the ability of a control system based on an incomplete or approximate model of the plant to achieve the desired result.

Feed forward control can serve a number of useful purposes. It may be impossible, or simply impractical, to measure the current state of the plant. It may not be obvious what error correcting signal should be applied as a result of the feedback that is received. Delays in the feedback path may also result in instability within high-gain feedback loops. In many cases open loop control is good enough to meet the required performance criteria.



(a) An open loop control system. (b) A closed loop control system.

Figure 2.2: Block diagrams of two different control systems. $R(t)$ is an input reference signal, $G(t)$ denotes the transfer function of the plant, $C(t)$ is the controlled output signal, $H(t)$ denotes an arbitrary feedback law, and $E(t)$ is the error signal.

Control Policy

A **control policy** is a mapping between a system state and the action to perform in order to accomplish a particular goal or terminal state. The general control policy can be represented as $\pi(\boldsymbol{x}, t)$. A control policy is often referred to as simply a policy or as a **controller** in the control systems literature.

An **optimal control policy** is denoted $\pi^*(\boldsymbol{x}, t)$ and is defined such that from any given state $\boldsymbol{x}$ the optimal control policy will reach the desired terminal state or follow a trajectory in a way that minimizes a quantity such as energy or time.

For under-actuated systems there are states from which it will be impossible to reach a desired terminal state or to track a given trajectory. For example, if one of the bipeds used in this thesis falls over it may lack sufficient torque to stand back up again. For these types of systems we define the set of states from which it is possible to reach the terminal state or track a desired trajectory as the **controllable region**.

Forward and Inverse Dynamics

There are two general formulations of the equations of motion for a bipedal robot: the Lagrangian formulation and the Newton-Euler formulation. The

Lagrangian equations of motion can be written as

$$\Gamma = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + N(\theta) + A^{\mathrm{T}}\lambda \tag{2.1}$$

where $M(\theta)$ represents the inertia matrix of the articulated figure in the current pose θ, C is the matrix of Coriolis and centrifugal forces, N contains gravity terms, A is the constraint matrix, λ contains the corresponding Lagrange multipliers, and Γ are the generalized forces. θ are the generalize coordinates, $\ddot{\theta}$ represents the current joint angular accelerations and $\dot{\theta}$ are the joint angular velocities.

The Lagrangian equations of motion have the advantage that the internal forces of constraint need not be explicitly represented in order to determine the motion of the robot. However, in general, the Newton-Euler formulation is computationally the most efficient with the computation time growing linearly with the number of degrees of freedom. For a discussion of this formulation applied to articulated skeletons see [4, 11, 12, 16].

Forward dynamics is the application of these equations to calculate the motion generated by a given force or torque. This is what our simulations use. The accelerations are integrated to update the system state. **Inverse dynamics** methods calculate the forces that would generate a given motion as defined by a set of accelerations.

Online Learning Algorithms

Many types of function approximators find approximate solutions to large systems of equations. These algorithms would be characterized as **offline** or memory based algorithms, as they require all of the data to be present in order to generate a solution. This is typical of parametric methods such as standard linear regression.

By contrast **online learning algorithms** do not need to store all of the data points but can perform incremental updates of their approximate solution as each new data point arrives.

Online algorithms are preferable in many situations because the models can be easily updated, they are adaptive to a slowly changing system, and are a more plausible model for human motor control. The online algorithms do not necessarily produce an identical solution to the batch algorithms, because without retaining all of the data certain error minimization criteria such as leave-one-out cross validation can only be approximated.

2.2 Imitation-based Learning

Humans learn to perfect skilled tasks through practice. Each attempt at performing a task provides information which updates an internal model leading to improved performance in the next trial [10]. The computational equivalent of this is the field known as reinforcement learning [46].

If we model the skilled task as a Markov Decision Process (MDP) or partially observable Markov Decision Process (POMDP) there are many algorithms for finding near optimal solutions to the control problem [6, 46]. When the dynamics of the MDP are not known in advance the parameters of the MDP typically need to be learned from observations of the system.

State-of-the-art algorithms such as E^3 [20] guarantee near-optimal performance can be obtained in time polynomial in the number of states of the system. The E^3 algorithm forces exploration of poorly modeled states in order to gain sufficient accurate statistics to determine state transition probabilities. In applications such as robotics where many states would lead to falls this seems wasteful in terms of computational resources. The dimensionality of the robot control problem also makes the polynomial time limits of the algorithm nearly

intractable for practical purposes.

With readily abundant kinematic descriptions of tasks, available as motion capture data, we wonder if these can be used to replace or speed up the learning of skilled tasks. As noted in Atkeson and Schaal [3] attempts to learn balance related tasks by replaying recorded human motion trajectories will fail. They identify many reasons for the failure of merely replaying trajectories including an imperfect inverse dynamic model of the robot arm, the task is slightly different (i.e. the robot grip of the pendulum is different than the human grip), and unstable tasks like balancing require feedback control.

Abbeel and Ng [1] describe an algorithm which avoids the aggressive exploration of state space and utilizes an initial teacher demonstration of the task. Rather than employing exploration moves, they concentrate on exploiting the dynamics learned so far. The basic algorithm is outlined below.

1. Have a teacher demonstrate the task to be learned. Record all state-action trajectories of the demonstration.

2. Use all state-action trajectories seen so far to learn a dynamics model of the system. For this model, find a near-optimal control policy using any reinforcement learning algorithm.

3. Test the policy by running it on the real system. If the performance is as good as the teacher's performance stop. Otherwise, add the state-action trajectories from the (possibly unsuccessful) test to the training set and go back to step (2).

The action taken at each step is represented by the joint torques. The state could either be the full body state or a projection such as the $(d, \dot{d})$ projection used in this thesis. This method has the benefit that at each evaluation of the policy the algorithm is making its best attempt to solve the problem.

Aside from reinforcement learning, we can also cast methods such as space-time constraints [58] as an imitation-based learning algorithm. The space-time constraints algorithm is a numerical optimization procedure which creates motion that respects physical laws. The minimization criteria is usually specified as minimum $\mathbf{x}$, where $\mathbf{x}$ is one of jerk, energy, or torque depending on the type of motion required. We can consider this technique as imitation-based learning since it is always seeded with an initial trajectory, usually from motion capture data, that provides the initial guess at the solution.

2.3 A Review of Walking

The synthesis of walking motion can be roughly broken into three broad categories: purely kinematic methods, hybrid kinematic-dynamic methods and purely dynamic methods. This thesis focuses on purely dynamic methods. While a complete review of all methods is virtually impossible, the popular and widely used techniques from each are briefly described below.

Kinematic based methods of walking

Purely kinematic descriptions of motion are widely used in computer animated films and video games. In kinematics only the description of motion trajectories is explicitly defined, there is no attempt to ensure that any of the motion trajectories respect Newton's laws of motion (Equation 2.1).

The first set of tools developed for motion specification were based on forward and inverse kinematics [32]. Forward kinematics consists of specifying the state vector of an articulated figure over time. This specification is usually done for a small set of key frames and interpolation techniques are used to generate in-between positions [24]. Defining key frames is usually left to a skilled animator, and the quality of motion is highly dependent upon their skill.

The use of forward kinematics makes it difficult to apply constraints to a motion, such as ensuring that the feet don't penetrate the ground. These constraints can be solved with inverse kinematic algorithms [57]. The relationship between the main task ΔX as expressed in Cartesian coordinates and the angular displacements $\Delta\theta$ takes the form

$$\Delta X = J\Delta\theta \tag{2.2}$$

where J is the Jacobian matrix of the system. The Jacobian is often not directly invertible, leading to a family of solutions that maintain the constraints.

Methods for synthesis of walking gaits based on bio-mechanical information were first introduced by Zeltzer [59] who used finite state machines parametrized by step length and velocity. The finite state machines generate key poses and linear interpolation is used to generate the in-between frames.

Methods for avoiding penetration of the ground with the stance foot were introduced by Bruderlin and Calvert [7] who use an inverted pendulum model for computing realistic gates. The root of the model is changed to be the current stance foot during the simulation and its position is fixed in the world frame.

Kovar *et al.* [22] introduce yet another technique that allows for the stretch of leg bones in order to enforce constraints on the feet to remove artifacts introduced during blending and interpolation of the in-between frames.

In addition to the specification of key frames by an animator, motion can be represented by directly capturing human performance. Motion capture systems use magnetic or optical technologies that record the global positions of markers in space during the performance of motion. A post-processing optimization step computes the pose of a hierarchical skeleton with respect to the marker positions for each frame of captured motion. Advances in motion capture technology have made it possible to easily capture thousands of motions [50].

For most applications the resulting captured motion needs to be modified in order to be useful. This may include some of the clean up techniques noted above to enforce foot placement constraints or simply to trim frames from the recorded motions to provide appropriate blend points.

Once a large motion capture database is generated there are numerous techniques for generating connections between motions. Motion graphs [21] compute the distance between each frame of motion in the database and generate a directed graph of connections between frames that are below some distance threshold. The resulting graph can be traversed in a manner that generates motion that was not originally captured.

Motions can also be described as a hybrid system of multiple local linear dynamical systems (LDS). In [25] motion clips are turned into linear dynamic systems called motion textons. These LDSs are then blended together at their end points to synthesize new longer motion sequences.

Hybrid Kinematic-Dynamic methods

All of the kinematic methods described previously have no guarantee that the resulting motions respect physical laws. They are primarily concerned with creating a particular look and feel of motion, which is important in the entertainment industry.

If the goal is to add realism or a certain amount of dynamics into motion then there are hybrid kinematic-dynamic methods that may be employed to add realism.

Space-time constraints [58] which were mentioned previously would fall in to this category. Given an initial motion trajectory, a numerical optimization operation is performed to ensure that defined constraints are respected. These can include modifying a jumping motion to leap a greater distance [45] or adding

a limp to a character by restricting motion in one joint [15].

In addition to global modifications that are performed by space-time constraints, small modifications can be added, particularly in interactive environments. Zordan and Hodgins [60] layer rigid body dynamics on top of motion capture data to allow captured motion to appear to react to pushes and hits.

Dynamic Walking

Kinematic and hybrid methods are very useful for generating realistic looking motions. The translation of these motions onto real robotic systems is not guaranteed to work, however. The generation of real dynamic motion is an extremely challenging problem.

Recent advances in biped walking robots have shown that techniques such as zero-moment point (ZMP) control [18] is an effective technique for creating real motion in robots. This is the technique which is used for the Honda Asimo robot [49]. ZMP techniques generally require flat footed robots, and generate slow moving trajectories. A feedback based balance control is also usually also employed. This tends to result in the "bent knee" stance of these robots to avoid singularity problems in the inverse kinematic algorithms used to maintain balance. The result is a rather unnatural looking walk cycle.

Passive dynamic walking is based on the natural dynamics of a walk cycle [26, 27]. A three dimensional passive dynamic walking robot with knees can walk on its own, requiring only a small amount of energy input to overcome energy lost due to friction with the ground [8].

Controllers, or control policies, aim to animate simulated figures or real robots with forward dynamics. This automatically accounts for the interactions with the environment. The difficulty is in creating the necessary torques that will make it perform the desired motion [9, 19].

Chapter 3

Dynamic Simulation

3.1 Overview

This chapter describes the details of the dynamical simulation which we use to model the various simple robot systems that we experiment with. The simulation of physical systems is a major area of research on its own. There are many choices and compromises that must be made with regard to numerical integration techniques, ground model interactions, and required precision of results. As presented earlier in Chapter 1 the biped simulations have the block structure shown in Figure 3.1.

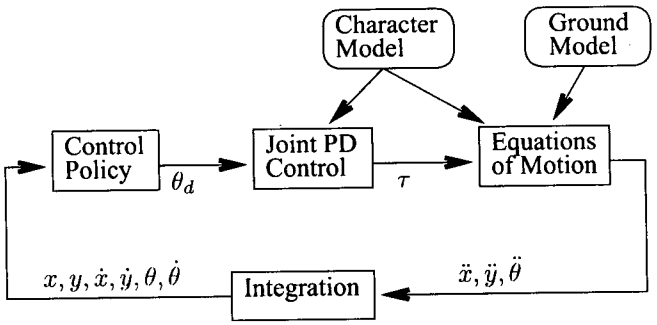


Figure 3.1: A block diagram of the simulation loop. Several significant components are represented, including the equations of motion for the biped, the ground interaction forces, the control policy in use and the various feedback loops involved.

link	parent	attach x	attach y	mass	inertia	mass x	mass y
1	0	0.0	0.0	70.0	1.475	0.0	0.0
2	1	0.0	0.0	5.0	0.0885	0.0	-0.225
3	2	0.0	-0.45	4.0	0.0696	0.0	-0.225
4	1	0.0	0.0	5.0	0.0885	0.0	-0.225
5	4	0.0	-0.45	4.0	0.0696	0.0	-0.225

Figure 3.2: An example of a physical description of a biped. This description corresponds to the 5-link biped used in some of the experiments. The *attach* x and *attach* y columns represent the joint position in the parent frame coordinates; the *inertia* column is the thin rod inertia scalar; and the *mass* x and *mass* y columns represent the coordinates for the center of mass of the link.

3.2 Equations of Motion

In an articulated figure with n links, the state vector $\{x, \dot{x}, y, \dot{y}, \theta_{1:n}, \dot{\theta}_{1:n}, c_{\text{left}}, c_{\text{right}}\}$ represents the full body state. The variables $x, \dot{x}, y, \dot{y}$ represent the global position and velocity of the root link of the articulated figure. Global orientation and angular velocity of the root link are described by θ_1 and $\dot{\theta}_1$. The variables $\theta_{2:n}$ and $\dot{\theta}_{2:n}$ represent angular positions and velocities relative to the link's parent frame of reference. The variables c_{left} and c_{right} are discrete Boolean variables representing contact of the left and right foot, respectively, with the ground.

A dynamics compiler [34] takes a description of the biped structure and parameters and produces C code for the equations of motion using the Newton-Euler equations of motion. A linear system having $n + 2$ equations and $n + 2$ unknowns solves for the unknown accelerations at each time step. An example description is shown in Figure 3.2.

The accelerations produced by the equations of motion are integrated using a simple forward Euler scheme. More sophisticated numerical techniques such as a 4th order Runge-Kutta integration scheme could be employed, though the very small time step imposed by the high gains in the ground reaction force would likely not produce much advantage to using a more sophisticated method.

3.3 Joint Proportional-Derivative Control

The joints of the articulated figure are driven by proportional-derivative control laws. A desired target angle and angular velocity is specified. When attempting to match a particular pose, rather than follow a trajectory, the angular velocity is specified as 0 rad/s. For this case, a PD controller can be visualized as a virtual spring and damper acting in parallel to pull a link to a desired angle from its current angle. A graphical depiction of this arrangement is shown in Figure 3.3. A joint torque is generated according to the following equation

$$\tau = K_p(\theta_d - \theta) - K_d(\dot{\theta}_d - \dot{\theta}) \tag{3.1}$$

where τ is the torque, K_p is the proportional gain, K_d is the damping factor, θ_d is the desired joint angle, $\dot{\theta}_d$ is the desired joint angular velocity, and θ and $\dot{\theta}$ represent the current angle and angular velocity of the joint.

The quality of control is determined by the magnitudes of K_p and K_d. High gain values imply quick response, but with the risk of overshooting the desired target angle, resulting in oscillation about θ_d. Small gain values, on the other hand, result in failing to reach the target angle due to lack of torque.

3.4 Ground Model

A significant aspect of the simulation of walking is the computation of ground reaction forces. There are numerous ways to compute collision and contact forces. We use a simple penalty method in our work. A set of points, called **monitor points** are defined on the feet of the biped. When one of these monitor points penetrates the ground plane ($y < 0$), an external force is applied trying to drive the monitor point back towards the ground plane. The force applied to the monitor point, M is computed as

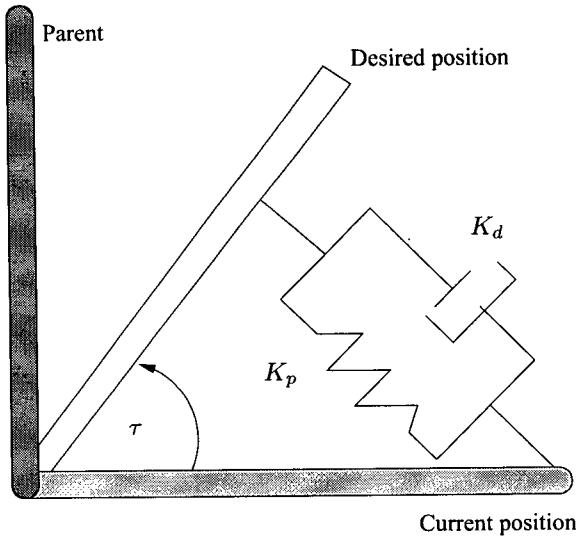


Figure 3.3: The effects of the proportional-derivative (PD) control law for a joint. The link is pulled towards the desired position with a torque τ that is proportional (K_p) to the angular displacement with the motion slowed by the damping factor K_d.

$$F = K_p(P - M) - K_d(\dot{M}) \tag{3.2}$$

where P and M are as defined in Figure 3.4.

Due to the high gains involved in simulating contact with the ground a small time step is required to ensure the Courant-Friedreichs-Lewy condition (CFL condition) is met and the simulation does not become numerically unstable. The fast simulation times of the two dimensional bipeds is not impacted by this small time step, though in a more complicated three dimensional simulation it is highly likely that a more sophisticated collision model would be required to avoid excessively slow simulations. Control torques are calculated every 0.005 s (200 Hz), while the time step of the simulation is 0.0001 s (10,000 Hz).

Other models for ground reaction forces include constraint based meth-

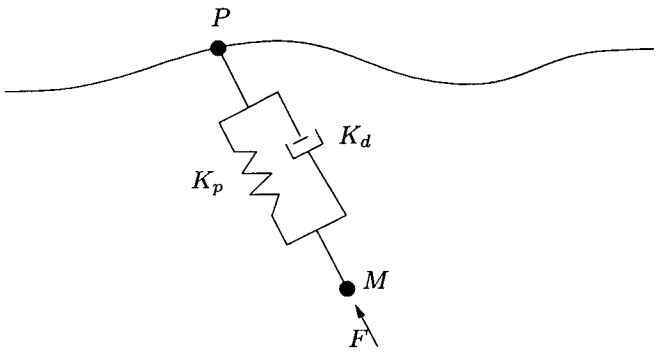


Figure 3.4: The model of ground interaction forces used in the dynamic simulations. The monitor point M is driven towards the entry point P with a force F as if it was connected with a spring of gain K_p and a damper of magnitude K_d.

ods [36], impulse based methods [29], and hybrid methods which handle initial contact with an impulse based method followed by the use of additional constraints during simulation [28].

Chapter 4

Locally Weighted Learning

This chapter provides an overview of locally weighted control policies and locally weighted learning, as well as the detailed workings of the specific locally weighted learning technique that we shall use in our control experiments. We are concerned with learning a control policy for biped walking. The control policies we choose to work with are of the general form

$$\hat{\boldsymbol{\theta}} = \pi(\boldsymbol{x}, t)$$

where $\hat{\boldsymbol{\theta}}$ is the set of desired joint angles, $\boldsymbol{x}$ represents the state vector of the biped, or a projection of the state space such as $(d, \dot{d})$, and t is time.

The joint angle trajectories of a walk are generally smooth non-linear functions of time, with the exception of a few points of discontinuity introduced during foot-ground interactions. We will model the control policy $\pi(\boldsymbol{x}, t)$ as a smooth non-linear function. There are numerous non-linear function approximation algorithms that could be used to model $\pi(\boldsymbol{x})$ in a supervised learning setting. We have set ourselves the following criteria:

1. free of negative interference during the learning process;
2. requires little knowledge of problem structure prior to learning;
3. fast to compute both in terms of learning speed and in evaluation;
4. capable of incremental learning;

5. scalable to a high number of dimensions;
6. ability to ignore irrelevant input dimensions.

Criteria (3) and (4) immediately eliminate a number of offline and memory based algorithms, restricting our search for learning algorithms to those that have online or incremental versions. Criterion (1) implies that we are looking for an algorithm capable of building local models, rather than trying to perform some sort of global function fitting to the training data. Criterion (2) implies a system that is structurally adaptive. A resource allocating network (RAN) of radial basis functions (RBF) would be one possible solution [35]. The method is constructive, so it only models areas where training data exists. The initial size of the RBF covers the entire input space and is optimized as more training data is presented. The resulting slow convergence [38] means that RAN fails to meet criterion (3). Criterion (6) suggests that some sort of principle component analysis (PCA) or other dimensionality reduction technique is required.

With the above considerations, our search is narrowed to non-linear function approximators that are structurally adaptive, based on local models and that can operate in high dimensions. These include Locally Weighted Principal Component Analysis [41], Locally Weighted Factor Analysis [55], and Locally Weighted Partial Least Squares [56].

We chose Locally Weighted Partial Least Squares. The incremental online version of LWPLS is known as Locally Weighted Projection Regression (LWPR) [53]. This is a sophisticated function approximation scheme that builds local linear regressions of a non-linear function. It is capable of operating in an online-fashion, including certain optimizations that make learning from trajectories (i.e. temporally coherent data points) extremely fast. The underlying partial least squares algorithm is also particularly appropriate to learning motion as it is based on the correlation between the input state and the output

target joint angles.

In prior work locally weighted regression techniques have been applied to learning "devil-sticking" [39], pole balancing [40] and to approximate the inverse dynamics model of a 7 degree-of-freedom robot arm [42]. These are all examples of fully actuated systems. The closest work to our own is found in [30] which decomposes a walk cycle into two steps and learns a target trajectory for each step based on a Poincare-Map projection of the state of a 5-link biped. In that work a set of via-points are learned that define a trajectory over half the walk cycle. This model lacks the continuous feedback that is present in our system. The learned control policy is also incapable of starting to walk, and Morimoto *et al.* use a manually initiated step to start the cycle.

The remainder of this chapter outlines the details of the LWPR algorithm.

4.1 Locally Weighted Projection Regression

The LWPR algorithm is an extension of Receptive Field Weighted Regression [37] which builds local linear regression models of non-linear functions. The name receptive field was coined by Schaal and Atkeson to reflect the biological inspiration for areas of local support in sensorimotor function [13].

A receptive field measures the relevance of a data point with respect to the current model using the Mahalanobis distance:

$$d_M(\boldsymbol{x}, \boldsymbol{c}) = \sqrt{(\boldsymbol{x} - \boldsymbol{c})^{\mathrm{T}} \boldsymbol{D} (\boldsymbol{x} - \boldsymbol{c})} \tag{4.1}$$

where $\boldsymbol{D}$ is a symmetric positive definite learned distance matrix, $\boldsymbol{x}$ is the query point, and $\boldsymbol{c}$ is the center of the receptive field.

A diagonal distance matrix $\boldsymbol{D}$ effectively represents the dimensions of an axis aligned hyper-ellipse, where all points on a given hyper-ellipse are considered

equidistant to the center. Put in another way, it allows for a relative weighting of various dimensions before a Euclidean distance from the center is computed. A distance matrix that is not diagonal (i.e. has off-diagonal elements) corresponds to an arbitrarily rotated hyper-ellipse. It is possible for some diagonal elements of $\boldsymbol{D}$ to be zero, indicating that the corresponding input dimension has no relevance to the regression. The distance matrix is usually stored as an upper triangular matrix $\boldsymbol{M}$ such that $\boldsymbol{D} = \boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}$.

We convert the Mahalanobis distance d_M into a relative weight using a Gaussian kernel function

$$K(d_M) = \exp\left(-\frac{1}{2}d_M\right) \tag{4.2}$$

This Gaussian kernel defined by the Mahalanobis distance has an infinite support region. In practice, a threshold value such as $w_{\text{thresh}} = 0.001$ is used such that if $K(d_M) < w_{\text{thresh}}$ then the associated receptive field is not updated or used in prediction to enforce finite support.

Table 4.1 provides a glossary of all symbols used in the locally weighted projection regression algorithms.

A receptive field defines the area over which a local model is learned. In contrast to competitive learning scenarios like neural-networks, the local linear models are learned completely independently of each other. Each receptive field is defined by its center $\boldsymbol{c}$ and its distance matrix $\boldsymbol{D}$. As will be seen later, the distance matrix is optimized during the learning process (cf. Equation 4.5). Throughout the optimization process the center $\boldsymbol{c}$ of each receptive field remains fixed.

The activation weight for a receptive field with respect to a training or query point is calculated according to Equation 4.2. When new training data is encountered that fails to activate a receptive field by a given creation threshold,

Symbol	Definition
α	A meta-learning rate that affects updates to $\boldsymbol{M}$
β_0^0	Initial regression parameters for the learned local model
$\boldsymbol{c}$	Center of the receptive field
$\boldsymbol{D}_{def}$	The initial distance matrix for a newly created receptive field
$\boldsymbol{D}$	The distance matrix for the receptive field
λ	A forgetting factor, the last $1/\lambda$ data points affect the model
$\boldsymbol{M}$	Upper triangular decomposition of $\boldsymbol{D}$ such that $\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M} = \boldsymbol{D}$
MSE	Mean squared error of predicted values vs. training values
n	Number of data points used to train receptive field
$\boldsymbol{p}_r^0$	Initial residuals from univariate projection r
$\boldsymbol{p}_r^i$	Residuals from univariate projection r after the i'th data point
r	Index for the current projection
R	Total number of univariate projections in the regression
$\boldsymbol{u}_r^0$	Initial direction for univariate projection r
$\boldsymbol{u}_r^i$	Direction for univariate projection r after the i'th data point
w	Weight of current training point
W^0	Initial average weight of data points used to train receptive field
W^i	Average weight of data points used to train receptive field after i'th data point
$\boldsymbol{x}$	Input vector used for training receptive field
$\boldsymbol{x}_0^0$	Initial mean input for receptive field
$\boldsymbol{x}_0^i$	Mean input for receptive field after the i'th data point
$\boldsymbol{x}_q$	A query point that we wish to predict an output value for
y	Output value used for training receptive field
y_q	A predicted output value for a query point $\boldsymbol{x}_q$
$y_{q,i}$	A predicted output value for query point $\boldsymbol{x}_q$ for the i'th receptive field

Table 4.1: A glossary of symbols used in the locally weighted projection regression algorithms

w_{gen}, a new receptive field is initialized according to Algorithm 4.1.

Algorithm 4.1 Initialize a receptive field

Input: Center for receptive field $\boldsymbol{c}$, default distance matrix $\boldsymbol{D}_{\text{def}}$.
Output: An initialized receptive field.

$\boldsymbol{c} \leftarrow \boldsymbol{c}$
$n \leftarrow 0$
$R \leftarrow 2$
$\boldsymbol{D} \leftarrow \boldsymbol{D}_{\text{def}}$
$\boldsymbol{x}_0^0 \leftarrow 0$
$\beta_0^0 \leftarrow 0$
$W^0 \leftarrow 0$
$\boldsymbol{u}_r^0 \leftarrow 0, r \in [1 \ldots R]$
$\boldsymbol{p}_r^0 \leftarrow 0, r \in [1 \ldots R]$

The prediction of the output based on a novel input point $\boldsymbol{x}_q$ is straightforward. Each receptive field provides its estimated output y_q as shown in Algorithm 4.2. These estimates are then combined in a weighted average according to the activation weight (Mahalanobis distance) for the query point with respect to each receptive field:

$$\hat{y}(\boldsymbol{x}_q) = \frac{\sum_{i=1}^{l} y_{q,i} K(d_M(\boldsymbol{x}_q, \boldsymbol{c}_i))}{\sum_{i=1}^{l} K(d_M(\boldsymbol{x}_q, \boldsymbol{c}_i))} \tag{4.3}$$

Algorithm 4.2 Predict with novel data

Input: An initialized and updated receptive field, a novel data point ($\boldsymbol{x}_q$)
Output: A prediction y_q

$y_q \leftarrow \beta_0$
$\boldsymbol{x}_q \leftarrow \boldsymbol{x}_q - \boldsymbol{x}_0$
for $r = 1$ to R **do**
  $y_q \leftarrow y_q + \beta_r \boldsymbol{u}_r^{\mathrm{T}} \boldsymbol{x}_q$
  $\boldsymbol{x}_q \leftarrow \boldsymbol{x}_q - \boldsymbol{u}_r^{\mathrm{T}} \boldsymbol{x}_q \boldsymbol{p}_r$
end for
return y_q

The overall training algorithm is presented in Algorithm 4.3. The model is initialized with no receptive fields, and is thus completely structurally adaptive. It only builds models in areas for which training data points exist. The LWPR

algorithm requires two parameters, $\boldsymbol{D}_{\text{def}}$, the default distance matrix that is used to initialize a new receptive field and w_{gen} the minimum activation energy. The initial number of projections is always initialized to 2. Input data should be scaled to have a zero-mean and a variance of 1 to ensure that the underlying partial least squares regression is valid.

Algorithm 4.3 Locally Weight Projection Regression

Input: A minimum activation weight w_{gen}, a default distance matrix $\boldsymbol{D}_{\text{def}}$
Output: A set of learned receptive fields

```
Initialize the LWPR with no receptive fields, K ← 0
for every new training point (x, y) do
  for k = 1 to K do
    Calculate activation with Algorithm 4.4
    Calculate current prediction error with Algorithm 4.5
    Update regression parameters and projections with Algorithm 4.6
    Update distance matrix, Equation 4.5
    Check if number of projections needs to increase, Equation 4.4
  end for
  if no receptive field activated by more then w_gen then
    Initialize a new receptive field with Algorithm 4.1
  end if
end for
```

The LWPR initializes itself with two initial projection directions, $R = 2$. Additional projection directions are added if the MSE at the next projection does not decrease more than a certain percentage of the previous MSE

$$\frac{\text{MSE}_{r+1}}{\text{MSE}_r} > \phi \tag{4.4}$$

where $\phi \in [0, 1]$.

The underlying regression model of LWPR is based upon partial least squares regression. An overview of the non-incremental version of PLS may aid in the understanding of Algorithm 4.5 and Algorithm 4.6. This thesis is concerned with the application of this technique to a learning problem, rather than the invention or modification of the learning technique. If the reader is not interested

Algorithm 4.4 Compute activation and update the means

Input: An initialized receptive field, a training point $(\boldsymbol{x}, y)$
Output: Activation weight for this receptive field, updated means

$w \leftarrow \exp\left(-\frac{1}{2}\sqrt{(\boldsymbol{x}-\boldsymbol{c})^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{x}-\boldsymbol{c})}\right)$
$W^{n+1} \leftarrow \lambda W^n + w$
$\boldsymbol{x}_0^{n+1} \leftarrow (\lambda W^n \boldsymbol{x}_0^n + w\boldsymbol{x})/W^{n+1}$
$\beta_0^{n+1} \leftarrow (\lambda W^n \beta_0^n + wy)/W^{n+1}$

in the actual mathematics this explanation can be skipped and they can skip ahead to Chapter 5.

Partial Least Squares (PLS) [14] builds a set of linear combinations of the inputs for regression. A univariate regression coefficient is first calculated on each dimension of the input training data $\boldsymbol{x}$. From this a derived input is constructed, which is the first partial least squares direction. The output $\boldsymbol{y}$ is regressed on this derived input. The input data $\boldsymbol{x}_1 \ldots \boldsymbol{x}_n$ is then orthoganalized with respect to the projection direction. The process is repeated for $m \leq d$ directions, where d is the dimension of the input training data $\boldsymbol{x}$.

Algorithm 4.5 Compute current prediction error

Input: A receptive field with updated means, an activation weight w, a training point $(\boldsymbol{x}, y)$
Output: Prediction error e_{cv}

$\boldsymbol{x}_{\mathrm{res},1} \leftarrow \boldsymbol{x} - \boldsymbol{x}_0^{n+1}$
$\hat{y} \leftarrow \beta_0^{n+1}$
for $r = 1$ to R **do**
  $z_r \leftarrow \boldsymbol{x}_{\mathrm{res},r}\boldsymbol{u}_r / \sqrt{\boldsymbol{u}_r^{n\mathrm{T}}\boldsymbol{u}_r^n}$
  $\hat{y} \leftarrow \hat{y} + \beta_r^n z_r$
  $\boldsymbol{x}_{\mathrm{res},r+1} \leftarrow \boldsymbol{x}_{\mathrm{res},r} - z_r \boldsymbol{p}_r^n$
  $\mathrm{MSE}_r^{n+1} \leftarrow \lambda \mathrm{MSE}_r^n + w(y-\hat{y})^2$
end for

One of the significant advantages of LWPR over other learning models is that not only are the linear regressions optimized as new training data is added, but the distance matrix $\boldsymbol{D} = \boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}$ for each receptive field is also optimized using an approximation of leave-one-out cross validation:

Algorithm 4.6 Update the local model

Input: Updated statistics from Algorithm 4.5
Output: Updated local regression parameters

$\text{res}_1 \leftarrow y - \beta_0^{n+1}$
for $r = 1$ to R **do**
  $a_{zz,r}^{n+1} \leftarrow \lambda a_{zz,r}^n + w z_r^2$
  $a_{zres,r}^{n+1} \leftarrow \lambda a_{zres,r}^n + w z_r \text{res}_r$
  $\beta_r^{n+1} \leftarrow a_{zres,r}^{n+1} / a_{zz,r}^{n+1}$
  $\text{res}_{r+1} \leftarrow \text{res}_r - z_r \beta_r^{n+1}$
  $\boldsymbol{a}_{xz,r}^{n+1} \leftarrow \lambda \boldsymbol{a}_{xz,r}^n + w \boldsymbol{x}_{\text{res},r} z_r$
end for
for $r = 1$ to R **do**
  $\boldsymbol{u}_r^{n+1} \leftarrow \lambda \boldsymbol{u}_r^n + w \boldsymbol{x}_{\text{res},r} \text{res}_r$
  $\boldsymbol{p}_r^{n+1} \leftarrow \boldsymbol{a}_{xz,r}^{n+1} / a_{zz,r}^{n+1}$
end for
$e = \text{res}_{R+1}$

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \tag{4.5}$$

where the cost function to be minimized is:

$$J = \frac{1}{W} \sum_{i=1}^{K} w_i (y_i - \hat{y}_{i,-i})^2 + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \tag{4.6}$$

where $\hat{y}_{i,-i}$ denotes the prediction of the model as if it were trained without the data point i, N is the total number of input dimensions, and K is the number of data points seen by the receptive field. The second term of this cost function ensures that as receptive fields see more data points they don't shrink to encompasses a single data point.

The implementation of LWPR is complicated, as there are several optimizations that may be applied to ensure that computational resources are minimized. The software used in this thesis was the reference implementation provided by Vijayakumar *et al.* [54]

Chapter 5

Imitation-based Learning Experiments

In this chapter we provide a detailed description of the experiments that were performed in the course of this thesis. Each experiment is designed to explore one or more of the goals presented in Chapter 1. The complexity of the experiments progressively increases as each builds on knowledge gained from the previous one.

5.1 Supervised learning of 3-link biped walking from observation of human control

The first experiment in imitation learning deals with the simplest configuration that is capable of walking. This is a 3-link biped that consists of a pelvis with a concentrated mass and two fixed length legs that are connected with pin joints to the pelvis. The model is shown in Figure 5.1.

Goal

The purpose of this experiment is to determine if the locally weighted learning infrastructure discussed in Chapter 4 is capable of reproducing a control strategy from observations of human input. Given a series of successful trials of human

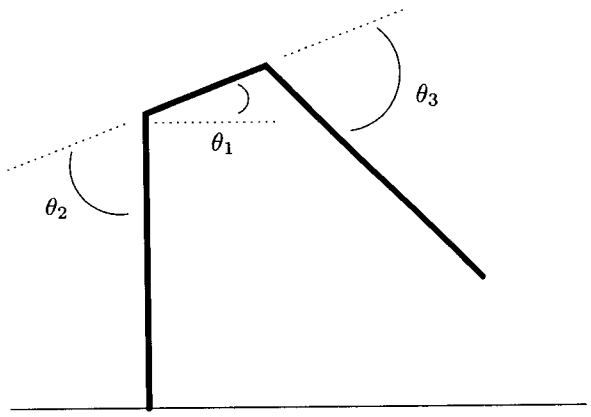


Figure 5.1: The 3-link walking biped. Two straight legs are connected to the hip with pin joints. Angles are measured relative to the parent frame, as noted by θ_1, θ_2, and θ_3. The legs for this model are 1m long, the hip is 0.3m wide. Most of the mass is carried in the hips, 50kg, while each leg has a mass of only 1kg.

control of a 3-link walking biped, can a control policy be learned? We apply this to one of the simplest possible bipedal robot structures that can walk, a 3-link biped.

Methodology

A 3-link biped simulation was created that was capable of operating in two modes. The first mode allows a human user to interactively direct the current target angles for the left and right hip joints of the robot using a mouse. These target angles are used by a PD control law (cf. Chapter 3) to generate torques in idealized motors in the hip joints resulting in a walking motion of the robot. The physical parameters of the simulation are outlined in Table 5.1.

A *trial* consists of an attempt to make the robot walk from a standing start a minimum of three steps without falling over. At any point in the simulation the human user can choose to save the *state history* for the current trial to a data file that will later be used to learn a control policy. The state history consists

Parameter	Value
pelvis width	0.3 m
leg length	1.0 m
pelvis mass	50 kg
leg mass	1 kg
hip K_p	1000 N/rad
hip K_d	100 Ns/rad
ground K_p	70,000 N/m
ground K_d	4,000 Ns/m

Table 5.1: Physical simulation parameters for a 3-link walking biped

Trial	Data Points
1	154
2	134
3	149
4	211
5	219
Avg	173
Total	867

Table 5.2: Details of trials for 3-link biped

of the horizontal distance from the stance foot to the center of mass of the robot and the horizontal velocity of the center of mass of the robot. This $(d, \dot{d})$ parametrization is a particular projection of the state space of the controller. In addition to this body state, the current action represented in terms of target angles for both hips are also stored. The sampling rate is 24 Hz.

After an initial training period the human user was able to consistently create a walking gait that traversed several steps without falling over. Five successful trials were recorded. Each trial represents data collected from the same user and represents between 2 and 5 successful steps. The details of each trial are listed in Table 5.2.

The data file with the five successful trials was used to generate a control policy in $(d, \dot{d})$ space with an output of target angles for the left and right hips. Locally Weighted Projection Regression (cf. Chapter 4) was used to

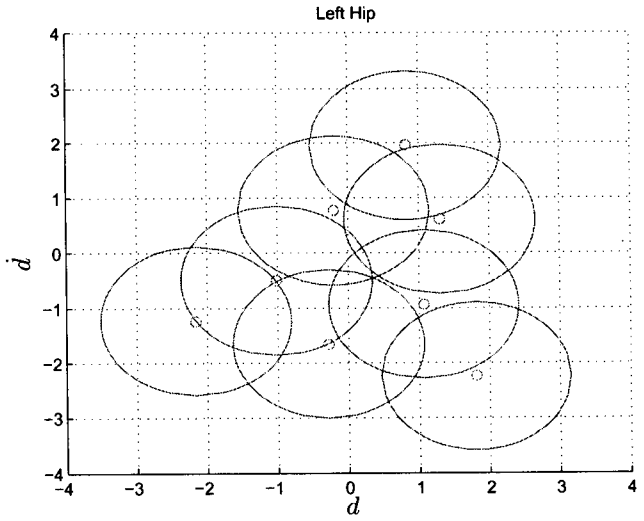
approximate the control policy. The control policy for the left and right hips were trained separately. Each policy has two input dimensions $(d, \dot{d})$ and one output dimension (θ).

$$\theta = \pi(d, \dot{d}) \tag{5.1}$$

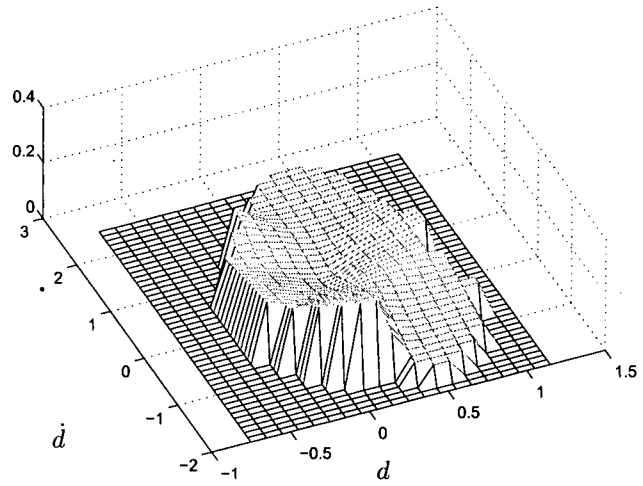
Results

The data file created during the human trials, as outlined in Table 5.2 was used to provide training points for the Locally Weighted Projection Regression algorithm. Each input dimension $(d, \dot{d})$ is independently zero-mean adjusted and scaled to have a standard deviation of 1. This scaling is important, as it allows some intuition to be applied regarding initial parameters for the size of the default distance matrix D used by the learning algorithm. The learning parameters for the LWPR algorithm are outlined in Table 5.3. The values chosen for the various learning parameters such as penalty, α, λ, and the initial $\boldsymbol{D}$ distance matrix are fairly typical. Some tuning of these parameters is done by hand, but this tends to be order of magnitude jumps in the parameters values and are generally related to the second derivative of the output with respect to the input $\partial^2 y / \partial \boldsymbol{x}^2$. That is, how rapidly the output function varies over the input space and how much support the local linear models should initially have.

The training phase consists of presenting each of the 867 data points to the LWPR algorithm exactly once. At the end of the training phase the control policy is represented by seven receptive fields for each of the left and right hips. The resulting policies are shown in Figure 5.2 and Figure 5.3. The resulting control policy successfully generates a walking cycle for the 3-link biped that does not fall over. The walk cycle repeats indefinitely over flat terrain.

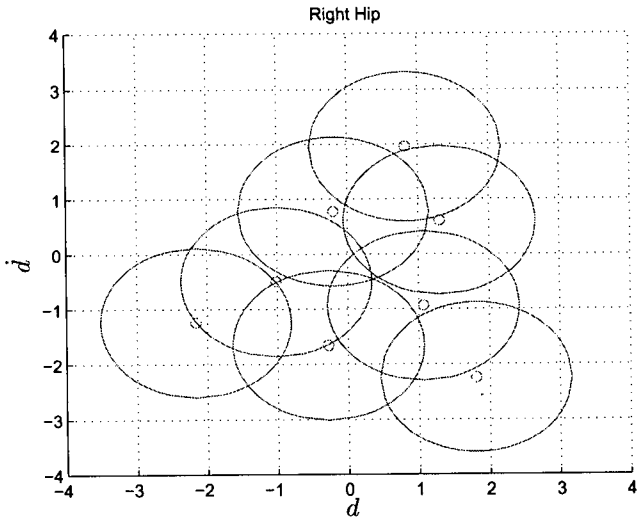


(a) Layout of receptive fields for left hip.

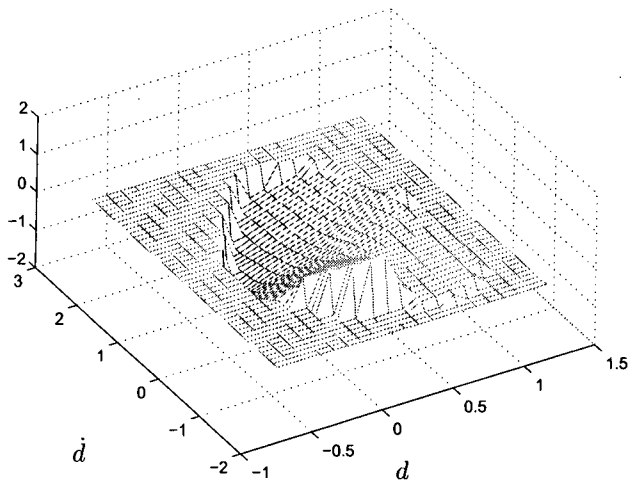


(b) Control policy for left hip.

Figure 5.2: The learned control policy for a 3-link planar biped. The large flat area related to an output target angle of 0° corresponds to areas outside the support of the receptive fields.



(a) Layout of receptive fields for right hip.



(b) Control policy for right hip.

Figure 5.3: A learned control policy for a 3-link walking biped. The flat areas related to an output angle of 0° corresponds to areas outside of the support region for the receptive fields.

Parameter	Value
diagonal only	true
meta learning	true
meta learning rate	100
penalty	1.0×10^7
initial α	10.0
initial λ	0.99
initial D	2.50**I**
prediction cutoff	0.001

Table 5.3: LWPR parameters for learning 3-link biped walking

Discussion

This addresses some of the goals for our work:

- Locally weighted learning can be used to control at least some under-actuated systems;
- Relatively few data points are required to generate an initial policy.

The results of this experiment were very positive compared to a simple nearest-neighbor implementation of a control policy. If the training data are stored and the nearest point in state space $(d, \dot{d})$ is used to look up a set of target angles, the 3-link biped fails to move from the starting position.

5.2 Supervised learning of 3-link biped walking policy based on full body state

Goals

The learning of the walking policy for the 3-link biped in Section 5.1 was based on a state space parametrization of $(d, \dot{d})$. Such a parametrization may not always be obvious for certain types of tasks, or it may not be measurable with

Parameter	Value
input dimensions	9
diagonal only	true
meta learning	true
meta learning rate	100
penalty	1.0×10^7
initial α	10.0
initial λ	0.9999
initial D	5.0**I**
prediction cutoff	0.001

Table 5.4: LWPR parameters for learning 3-link biped walking using full body state.

on-board robotic sensors. The goal of this experiment is to use an existing control policy to train a new control policy based on the full body state of the robot.

Methodology

The initial experiment is set up in the same manner as in Section 5.1. A control policy π_α is learned based on the five human trials. A second control policy π_β is defined with the parameters as outlined in Table 5.4. Note that some of these parameters, λ and $\boldsymbol{D}$ are different from the first experiment. The larger initial $\boldsymbol{D}$ value means smaller receptive fields. With smaller fields and higher input dimensions, each field is activated by fewer input training points. The higher λ value allows each receptive field to retain more knowledge of previous data points as part of it's local linear model. The key difference for this controller is that the input state is specified as the full body state $(\dot{x}, y, \dot{y}, \theta_{1:3}, \dot{\theta}_{1:3})$, which includes all dimensions, exclusive of absolute horizontal position.

The simulation is setup in an episodic learning environment. Each trial consists of allowing π_α to control the robot and walk to a terminal distance of 2m. The state history, sampled at 24 Hz, for the entire trial is used as training

data to the π_β control policy. After every 100 trials the control is switched to use π_β and the quality of the resulting policy is evaluated for 10 trials.

Results

The quality of the second control policy π_β was very dependent on the LWPR parameters shown in Table 5.4. Both the initial distance matrix $\boldsymbol{D}$ and the initial λ had a large impact on the learning rate of the control policy. With the parameters that are outlined in Table 5.4 the control policy π_β capable of generating a stable walk cycle was learned in 900–1100 trials.

Discussion

The higher dimensionality of the control policy, π_β, requires more training data to develop a robust and stable control policy. The LWPR algorithm was able to reject irrelevant dimensions and there were 4 projections per receptive field, on average. The $(d, \dot{d})$ projection used in the first experiment seems to imply that two projections should be enough to control the 3-link biped. However, d and $\dot{d}$ are measured relative to the stance foot, so during the walk cycle the values switch signs. This implies that during the two phases of the walk the stance hip angle and stance hip angular velocity are the important controlling variables.

The sensitivity to the initial parameters was surprising, as all other experiments were relatively immune to changes in these values. Further analysis should be performed to determine if inappropriate scaling of the training data was occurring, which would affect the underlying partial least squares regression. However, this experiment does show that if a suitable parametrization exists, such as our $(d, \dot{d})$ projection, where rapid learning can occur, then a supervised learning task in a simulation environment can be used to create a control policy based on a perhaps more general representation of the state.

5.3 Improving the 3-link biped walking policy

Having shown in Section 5.1 that the elements of the learning infrastructure we have created are capable of imitating a specific simple observed control policy, we are now interested in improving a policy that is represented as a set of receptive fields.

Goals

There is currently no guarantee that the observed policy is an optimal policy, or that it would even be a successful policy if the observations were of a slightly different configuration (e.g. different masses or leg lengths). We wish to use a policy search method to *improve* the existing policy to better meet some criterion. We chose to maximize the horizontal velocity of the walking cycle. The horizontal velocity for a given policy starting in state $\boldsymbol{x}$ is defined as

$$J^{\pi}(\mathbf{x}) = \frac{x}{T} \tag{5.2}$$

where T is the time taken to travel a distance x.

The optimal horizontal velocity starting from state $\mathbf{x}$ is denoted by $J^{*}(\mathbf{x})$, that is

$$J^{*}(\mathbf{x}) = \max_{\pi} J^{\pi}(\mathbf{x}) \tag{5.3}$$

Methodology

There are a large number of optimization algorithms that can be employed when attempting to improve a policy. The method chosen for this experiment was Stochastic Policy Gradient Descent [48]. The intuition for this method is straightforward. A small random vector is chosen to modify the parameters of a

system. The policy is evaluated, and if it improves the change is kept, otherwise the *opposite* change is applied. This latter step is not used in our case, as it tended to force the biped out of the *controllable region.*

At the beginning of each trial a random vector is drawn uniformly from the unit hypersphere. Each dimension of the vector is drawn uniformly from the unit hypercube. If the resulting vector lies within the unit hypersphere it normalized to lie on the surface of the hypersphere, otherwise it is rejected and each dimension is drawn from the unit hypercube again. This rejection sampling ensures that the direction of the perturbing vector is uniformly samples and is not biased towards the poles.

Each element of the vector represents a modification to the mean output value for a receptive field. For example, with seven receptive fields, a 7-dimensional vector is created $\Delta\pi_i \sim \mathcal{U}(-1,1)$ and the result is normalized $|\Delta\pi| = 1$. This normalized vector is then scaled by another value M which is drawn from a Gaussian distribution with a mean of 5 and standard deviation of 1: $M \leftarrow \mathcal{N}(5,1)$. This scale factor is clamped to a range of $[0,10]$ to keep the new policy guess close to the existing policy. The numbers represent degrees in our case. The clamping limits and the Gaussian distribution parameters were chosen through trial and error. They produce sufficient step changes to modify the behavior of the 3-link biped while remaining close to the existing policy.

Each time the control policy is queried, the output from each receptive field is modified according to the appropriate element of $\Delta\pi$. A trial is terminated if the 3-link biped falls over (failure), time expires $t > 10$ (failure), or a distance of $2m$ is traveled (success). If a trial is successful then the average horizontal velocity is calculated:

$$J_N^\pi = \dot{x}_N = \frac{x_N}{T_N} \tag{5.4}$$

where x_N is the terminal horizontal position and T_N is the time taken for the trial.

We let $\hat{J}$ be our current estimate of J^*. If $J_N^\pi > \hat{J}$ then the state history for the current trial is used as training data and applied to the LWPR representation, in exactly the same manner that the initial policy was trained. We present the state history as new training data, rather than modifying the regression parameters β_0 directly, so that the internal statistics of the learning algorithm are maintained in a consistent manner. This algorithm is outlined in Algorithm 5.1.

Algorithm 5.1 Improve A Policy

Input: An initial policy π and a maximum number of trials $T \geq 1$
Output: An improved policy π^*

$\pi_0 \leftarrow \pi$
$\pi^* \leftarrow \pi_0$
$J_0 \leftarrow \text{Simulate}(\pi_0)$
$\hat{J} \leftarrow J_0$
$N \leftarrow 1$
while $N \leq T$ **do**
  $\Delta\pi \sim \mathcal{U}(-1, 1)$
  $\Delta\pi \leftarrow \Delta\pi / |\Delta\pi|$
  $M \leftarrow \mathcal{N}(5, 1)$
  $\pi_N \leftarrow \pi^* + M\Delta\pi$
  $J_N \leftarrow \text{Simulate}(\pi_N)$
  if $J_N > \hat{J}$ **then**
    $\text{Train}(\pi_N)$
    $\pi^* \leftarrow \pi_N$
    $\hat{J} \leftarrow J_N$
  end if
  $N \leftarrow N + 1$
end while
return π^*

Results

Out of a total of five runs of the policy improvement algorithm, each consisting of $T = 100$ trials, the overall improvement of the policy is plotted in Figure 5.4. Each trial represents a new control policy π. The mean overall improvement

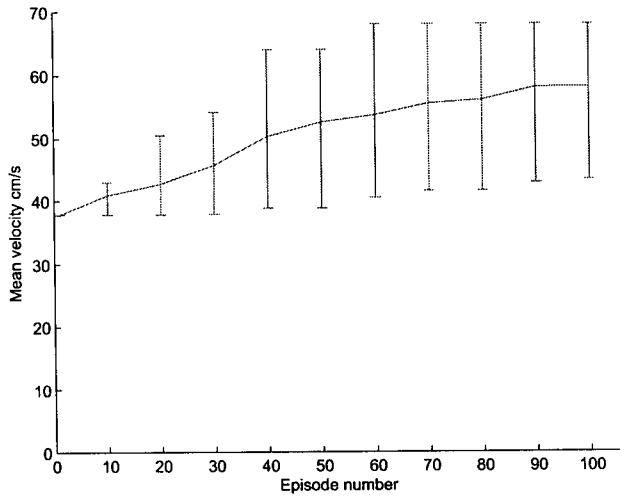


Figure 5.4: Policy improvement results from five trials consisting of 100 episodes each. The mean average velocity improves from 38 cm/s to 58 cm/s. Note the large differential between minimum and maximum on the individual trials, a result of the random nature of Stochastic Gradient Policy Descent

within 100 episodes is 52%. We should note the high variability, as shown by the min-max bars, for the individual trials. The random nature of the stochastic policy gradient descent algorithm requires numerous restarts to generate an *improved* policy.

Discussion

This experiment shows that locally weighted control policies that are initialized by one means can be improved with respect to particular criteria using a form of policy search. By presenting improvements as new training data we maintain the internal statistical representation of the learning models without having to directly manipulate the regression parameters.

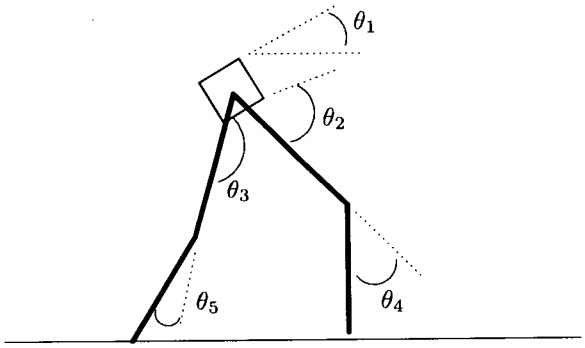


Figure 5.5: The 5-link walking biped. Two legs with knees are connected to the hip with pin joints. The thighs and calves for this model are 0.45m long. Most of the mass is carried in the hips, 70kg, while each leg has a mass of only 9kg. All angles are measured relative to the parent frame, as noted by $\theta_{1...5}$

5.4 Supervised learning of 5-link walking from finite-state-machine control observations

The results of Section 5.1 and Section 5.3 are encouraging. We now move on to experiments involving a less stable robot design, a planar 5-link biped. This robot has more degrees of freedom to control, as it has knees. The robot is very unstable, given its lack of ankles and so the only method of maintaining balance is to take steps. Figure 5.5 shows the model.

Goals

For this experiment we are looking to achieve the following goals. First, can the locally weighted policy technique support more degrees of freedom and deal with a more difficult problem. Second, can an existing control policy be learned from the direct expression of that policy?

Parameter	Value
pelvis width	0.1 m
thigh length	0.45 m
calf length	0.45 m
pelvis mass	70 kg
thigh mass	5 kg
calf mass	4 kg
hip K_p	300 N/rad
hip K_d	30 Ns/rad
knee K_p	300 N/rad
knee K_d	30 Ns/rad
ground K_p	70,000 N/m
ground K_d	4,000 Ns/m

Table 5.5: Physical simulation parameters for a 5-link walking biped

Methodology

The 5-link walking biped has a concentrated mass at the hips, and two legs with knees, as shown in Figure 5.5. The physical parameters are outlined in Table 5.5. The 5-link biped has an existing controller that is based on a pose control graph similar to the one developed in [52], and described below.

The training data is provided by an existing control policy that has a pose control graph structure of the form $\hat{\theta}_i = f_i(\boldsymbol{x})$ for the joints numbered $1 \ldots 4$. The joint numbering corresponds to the left hip, left knee, right hip and right knee. For the purposes of this thesis, the specific details of the reference functions $f_i(\cdot)$ are not relevant , as we are interested in learning control policies from partially observable systems.

The existing pose control graph controller is essentially an open loop control policy. Each state is defined by a target pose parametrized by desired speed, stride length, and velocity. State transitions occur either after a specified period of time has elapsed (timing states) or the swing foot contacts the ground (contact states). A total of four states exist, corresponding to the states left foot takeoff (LTO), left foot plant (LFP), right foot takeoff (RTO), and right foot plant

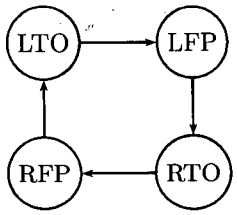


Figure 5.6: A walking cycle can be decomposed into four distinct phases: left foot takeoff (LTO), left foot plant (LFP), right foot takeoff (RTO), and right foot plant (RFP)

(RFP). This is shown graphically in Figure 5.6.

Initially we attempted to learn the control policy directly but this exposed a weakness in the LWPR framework. Discontinuities are not handled particularly well in the LWPR framework, as the distance optimization algorithm works aggressively to shrink receptive fields away from the discontinuities that exists at the state transitions.

This failure led to the creation of an enhanced pose control graph structure, which we term the *policy control graph.* The policy control graph, rather than having each node in the graph represent a pose, in our framework it represents a control policy. Termination criteria for the policies are time based for the left foot and right foot take off (LTO and RTO) phases, and contact based for the left foot and right foot placement phases (LFP and RFP). These are identical to those in the pose control graph, though the duration of the take off phase is slightly shorter in the policy control graph (0.2 vs. 0.3 seconds) as this provided a more robust walking controller.

Learning is now accomplished by sampling the pose control graph output over an expected range of the parametrized state space $(d, \dot{d})$. The original pose control graph expresses a desired pose, and in this implementation contains a small amount of feedback related to the stride length d and horizontal velocity $\dot{d}$. We uniformly sample across the expected range of $(d, \dot{d})$ and train a set of

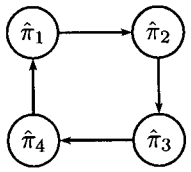


Figure 5.7: The policy control graph structure. Each phase of the walk cycle is now represented by an approximation of the original control policy and thus has its own distinct set of receptive fields.

Parameter	Value
diagonal only	true
meta learning	true
meta learning rate	15
penalty	1.0×10^7
initial α	10.0
initial λ	0.99999
initial D	25.0**I**
prediction cutoff	0.001
samples	100,000

Table 5.6: LWPR parameters for learning 5-link biped walking

receptive fields using the LWPR algorithm. The range is slightly larger than that observed to provide robust control in the existing pose control graph structure.

The control policy for each of the hips and knees is learned independently for each of the four poses represented in the original pose control graph. The LWPR learning parameters are shown in Table 5.6. These parameters are slightly different from those presented in previous experiments. No attempt has been made to find the lower limits to learning the pose control graph structure.

Results

The resulting walk cycle generated from the policy control graph learned using this technique produced walk cycles that were qualitatively indistinguishable from the original pose control graph. Different walk cycles can be generated

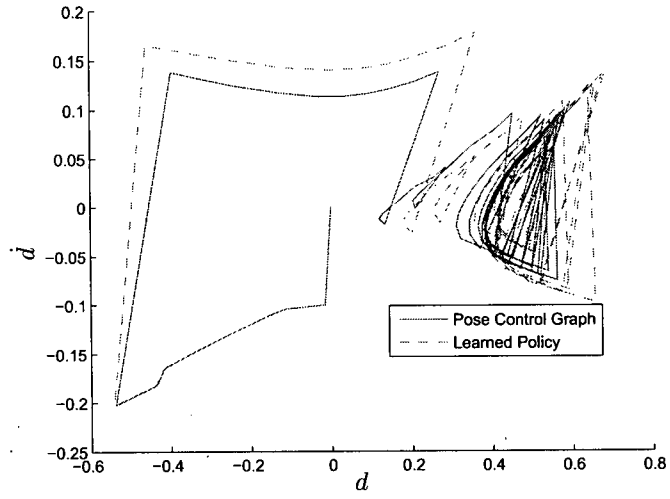


Figure 5.8: A comparison of a walk cycle generated from a pose control graph and the learned policy control graph

with horizontal velocities that range from -0.4m/s to 1.2m/s. Each distinct speed requires learning a different control policy with a new set of training data. A comparison of the pose control graph (original) walk cycle compared to the learned policy control graph is shown in Figure 5.8. Note the slight difference in the pose control graph output with respect to the learned policy control graph functions. The difference is attributable to the difference in take off phase duration, resulting in the learned policy control graph having a slightly faster gait than the original policy.

Discussion

It is not really surprising that the learning algorithm can replicate the smooth control provided by the pose control graph. What this experiment has shown is that the policy control graph structure is capable of controlling more complex

movements in unstable and underactuated systems. It has also highlighted the difficulties in trying to model systems with discontinuities. We have chosen to solve this problem through the use of a finite state machine, represented by the policy control graph, to distinguish between the distinct phases of the motion. Other methods, such as those proposed in [51] have also been proposed to automatically decompose an observed system into a set of models.

One limitation of the approach taken in this experiment is that we still need access to a working controller in order to collect the data. This situation is rare and led us to experiment with deriving a control policy from direct kinematic descriptions of data. We allowed the existing control law to run through 100 trials each of which had a random external force in the range of $[-20, 20]$ Nm/s^2 applied horizontally to the center of mass. Full state information was recorded along with the output torques for each joint and the phase of the walk cycle that the sample corresponded to. A policy control graph was trained using this data. The resulting walk cycle, though it was capable of making the planar 5-link biped walk, was not nearly as stable as the existing control law. This is likely due to poor modeling of the torques required by the stance leg's knee joint which must remain locked to prevent the robot from falling. The quality of the learned model is sensitive to the simulation time step and control sampling frequency. The method shows promise and will likely be explored in the future, using motion segmentation techniques such as Kinematic Centroid Segmentation [17] to derive phase information and inverse dynamics techniques [11] to compute required torques.

5.5 Learning to transition between different walking speeds on a 5-link biped

The preceding experiments have shown that it is possible to learn a skilled task, in this case walking, on different bipeds and with different means (existing control law, human observation, and kinematic description). A robot that can perform a single task, while useful in assembly plant operations, is not particularly interesting when considering a larger context of a robot that can interact with its environment.

Goals

This final experiment has two goals. The first goal is to determine if it is possible to switch between different control policy graphs at natural transition points. For example, can we switch from left foot takeoff (LTO) at one speed to left foot plant (LFP) at another speed, either faster or slower. The second goal is to automatically discover feasible transition points between existing control policy graphs. Given working control policies for different motions, how do we go about creating a probabilistic transition matrix between graphs? We wish to define a transition matrix that provides probability of successfully transitioning between two nodes of different policy control graphs.

Methodology

A family of 11 policy control graphs for biped walking were created, based on the results of Section 5.4. These walking controllers were parametrized by speed such that the average velocity generated by policy π_i is strictly less than the average velocity generated by policy π_{i+1}.

The first goal of the experiment is tested by allowing the user to set a desired

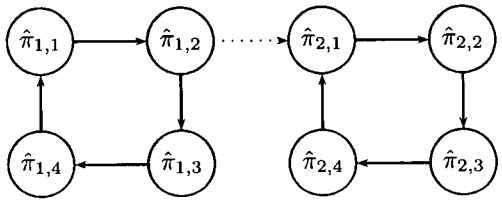


Figure 5.9: Transitioning between policy control graphs. The solid lines represent the usual transitions, successful with probability 1. The dotted line represents a test transition between the two different policy control graphs.

velocity interactively. At the termination of each node in the policy control graph the current velocity is compared to the desired velocity. If the desired velocity is less than the current velocity then the next slower control policy graph is chosen. If the desired velocity is greater than the current velocity then the next faster control policy graph is chosen. Only a single step in speed is allowed at each termination test.

For the development of the probabilistic transition matrix, the following technique is used. Transitions between nodes of a policy control graph occur when the active node reaches its termination criterion. Normally the next node would correspond to the next phase of the walk cycle. With a set of policy control graphs we wish to potentially transition to a node in a different policy control graph. A simplified example with just two policy control graphs is depicted in Figure 5.9. The dotted line represents a potential new transition.

A default transition matrix of dimension 55×55 was created. The 55 dimensions correspond to each of the 4 phases of the walk cycle, plus an additional start phase that corresponds to the beginning of a new trial and always transitions to left foot takeoff (LTO) when contact with the ground is made. These five phases exist for each of the 11 speeds of the policy control graph that we have generated. The normal probability transitions were encoded, such that the natural cycle of each policy control graph was maintained. That is, for each

policy π_i, there are four nodes, corresponding to the four phases of the walk cycle. Each node has a probability of 1 that it will successfully transition to the next node in the normal cycle. A portion of this default matrix is shown below. The fifth (start) phase has a transition probability of 1 to the LTO phase of its policy control graph.

$$T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 0 & \dots \\ 1 & 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & 0 & \dots \end{pmatrix}$$

A systematic exploration of transitions could be created, where test transitions are setup between all possible pairs of nodes in the graph. This would be thorough, but the simulations would have to be exhaustive in order to start populating the higher speed policies. Instead a random test connection is made so that some connections between all policy control graphs are examined early in the exploration phase.

Prior to each trial a random connection with a probability weight of 1 is made between two arbitrary nodes in the family of policy control graphs. The restrictions on placing this temporary connection were that nodes can not connect to themselves, and they can not connect to another node in their own policy control graph (i.e. we want to transition to a different speed walk). When selecting the next node, the transition matrix is examined and the next node is selected based on the cumulative probability within the transition matrix.

The simulations were setup to conduct trials, terminating when the biped fell down (failure), simulation time exceeded 10 s (success), or the robot traveled more than 5 m (success). Each trial started from the same initial stationary

pose. When a trial is terminated it is evaluated in terms of success or failure. If the temporary connection between policy control graphs was used at some point in the trial, and the trial was a success, then the appropriate entry in the transition matrix is incremented by 0.01. The simulations are allowed to run for 7,500 episodes to populate the values in the transition matrix.

Results

The interactive active aspect of this experiment produced very favorable results. The different policy control graphs are able to transition to the next natural phase in adjacent speeds at all phases. The model does not need to recover in any way, that is there is no stumble or disruption of the walk cycle. In this manner it is possible to have the robot walk backwards and forwards in the simulation essentially indefinitely.

For the development of the probabilistic transition matrix the following results were observed. When left to execute, a transition matrix is built up that supports transitioning between families of controllers. Some of the transitions are unstable, and while they do not produce a fall, end up putting the biped through a small recovery phase (which it is capable of doing due to its robustness). Think of stubbing your toe, you don't fall down unless on your recovery step you also stub your other toe. These unstable transitions are identifiable by a probability $p \leq 0.5$ when all values have been normalized. The 0.5 threshold is determined arbitrarily, to indicate that the robot fails to transition more often than it succeeds.

If left to run for too long, then too many possible transitions exist the next node selection can lead to hopping from family to family too often and the biped falls over. This could be changed through a modification of the transition law to favor same family transitions only until the user requests a different speed.

Empirically, it seems that more than 15,000 trials results in a probabilistic transition matrix that fails to produce a robust walk cycle.

Discussion

The results from this experiment were successful. The biped is capable of transitioning to both faster and slower walk cycles from every node of each policy control graph. The most successful transitions are those that branch to the next natural phase in an adjacent speed policy control graph. Random transitions to arbitrary nodes in any policy control graph are possible, though the robot requires at least one complete walk cycle to recover from any induced perturbations.

A limitation in extrapolating from the results of this experiment is that the motions that were part of the transition matrix were all very similar, that is they are all walk cycles of varying speeds. The addition of more types of motion such as walking up or down stairs or a hop would add considerable variation to the possible motions and yield more interesting results.

The receptive field implementation also allows for the inter-state transition to occur not just at termination of state nodes in the policy control graph but also in overlapping receptive fields of different motions. This would require all control policies to work with the same projection of the system states (i.e. $(d, \dot{d})$) or the full state $\boldsymbol{x}$. That is, if a jump is described using a different state parametrization it can still be tested for safe transitions with the method used in this experiment.

Chapter 6

Conclusions

As noted in the introduction, the study of motor control is a large interdisciplinary area of research. There are many ways to view the problem and equally as many avenues for trying to replicate skilled motion on robots. This thesis has explored a specific subset of approaches, based on imitation-based learning and locally weighted control policy representations. This thesis has not fully solved the problem of teaching a robot how to walk. However it has shown that recent advances in areas of machine learning can be successfully applied to this very challenging problem. The specific contributions of this thesis are outlined in the following section, followed by a discussion of future research.

6.1 Contributions

Locally Weighted Projection Regression

We have shown, through a series of experiments, that locally weighted projection regression is capable of learning control policies in under-actuated robotic systems. Previously this learning method had been primarily used to learn aspects of motor control to fully actuated systems (e.g. a robot arm). While similar in nature to [30], we have shown that higher degree of freedom robots, including those with knees, can be controlled with policies represented by LWPR.

Policy Control Graph

We are no longer limited to simple open-loop control systems, but can direct coordinated actions through the ability to transition between multiple control policies. Each node within the policy control graph represents a complete control policy that has full feedback information.

We have shown that transitions between control policy graphs may be defined at the node level. An approach to evaluating the probability of success of a transition was described, with the result being a probabilistic transition matrix indicating the likelihood that a particular switch in the active control policy graph will succeed. We also showed that transitions to the next natural phase in a walk cycle in adjacent speeds of policy control graphs are always successful. This opens the possibility of performing motion planning using the information regarding transition capabilities.

6.2 Future Work

Animation with Physics Based Characters

The extension to three dimensional models and simulation is an obvious piece of future research. The use of planar bipeds in this thesis greatly reduces the simulation time, making the exploration of ideas significantly faster. With the establishment of some of the fundamental ideas and the presence of a working learning infrastructure the extension to higher dimensions should be straightforward.

The use of higher degree of freedom models is also a next logical step. The 5-link biped used in some of the experiments has a total of four controllable degrees of freedom in a twelve dimensional state space. Character models used in video games have closer to twenty-seven controllable degrees of freedom in a sixty

dimensional state space. The use of LWPR should help, as the computational complexity has been shown to be linear in input dimensions, not exponential.

Improving Observation

In the experiment that learned a control policy from observed joint torque data we had access to the actual target joint angles produced by an existing control policy. This is generally not the case and we would like to extend our learning framework to support inverse dynamic calculations based on observed kinematic motion (i.e. motion capture data).

The use of techniques such as the Articulated Body Method [11], or the use of space-time constraints [58] to generate joint torques for an observed motion would provide us with a wealth of new training examples, accessible through public motion capture databases and our own motion capture facility.

Improving Optimization

The optimization techniques used in this thesis are fairly straightforward, and are not particularly fast when applied to higher dimensional problems. There exist sophisticated nonlinear optimization algorithms which could potentially be employed to improve the quality of the control policies. A technique such as Stochastic Meta Descent [43] is a very fast technique that works in high dimensions.

Learning motion of points of support

Balance is related to the center of mass of an articulated figure and the points of support. An exploration of how the points of support are placed in relation to the dynamics of the center of mass is worth investigating. The discontinuity of this function suggests that recent work in [51] might be required.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *ICML 05: Proceedings of the 22nd International Conference on Machine Learning*, pages 1–8, August 2005.

[2] Robert O. Ambrose, R. S. Askew, W. Bluethmann, M. A. Diftler, S. M. Goza, D. Magruder, and F. Rehnmark. The development of the robonaut system for space operations. In *ICAR 2001 Invited Session on Space Robotics*, August 2001.

[3] Chris G. Atkeson and Stefan Schaal. Learning tasks from a single demonstration. In *ICRA 1997: Proceedings of the 1997 IEEE International Converence on Robotics and Automation*, volume 2, pages 1706–1712, April 1997.

[4] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. In *Proceedings of the 15th annual conference on Computer graphics and Interactive techniques*, pages 179 – 188. ACM Press, 1988.

[5] Richard Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[6] D. P. Bertsekas and J. Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, 1996.

[7] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 233–242, New York, NY, USA, 1989. ACM Press.

[8] Steven H. Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *International Journal of Robotics Research*, 20(7):607–615, 2001.

[9] Steven H. Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

[10] Opher Donchin and Reza Shadmehr. Linking motor learning to function approximation: Learning in an unlearnable force field. In *Advances in Neural Information Processing Systems*, volume 14, pages 195–204. MIT Press, 2002.

[11] R. Featherstone. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[12] R. Featherstone and D. E. Orin. Robot dynamics: Equations and algorithms. In *ICRA 00: Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, volume 1, pages 826–834, April 2000.

[13] A. P. Georgopoulos. Higher order motor control. *Annual Review of Neuroscience*, 14:361–377, March 1991.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, New York, NY, USA, 2001.

[15] Eugene Hsu, Kari Pulli, and Jovan Popovič. Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089, 2005.

[16] Paul M. Isaacs and Michael F. Cohen. Controlling dynamic simulation with kinematic constraints. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 215–224, New York, NY, USA, 1987. ACM Press.

[17] Odest Chadwicke Jenkins and Maja J. Matarič. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 225–232, New York, NY, USA, 2003. ACM Press.

[18] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *ICRA 2003: Proceedings of the 2003 IEEE International Conference on Robotics and Automation, 2003*, volume 2, pages 1620–1626, 2003.

[19] S. Kajita, T. Yamaura, and A. Kobayashi. Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Transactions on Robotics and Automation*, 8(4):431–438, August 1992.

[20] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, November 2002.

[21] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM Press.

[22] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104, New York, NY, USA, 2002. ACM Press.

[23] Fritz Lang and Thea von Harbou. Metropolis, 1926.

[24] John Lasseter. Principles of traditional animation applied to 3d computer animation. *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer Graphics and interactive techniques*, 21(4):35–44, 1987.

[25] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472, New York, NY, USA, 2002. ACM Press.

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

[27] Tad McGeer. Passive walking with knees. In *Proceedings 1990 IEEE Robotics and Automation Conference*, pages 1640–1645, 1990.

[28] B. Mirtich. Hybrid simulation: combining constraints and impulses. Technical report, University of California, Berkeley, 1996.

[29] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff., New York, NY, USA, 1995. ACM Press.

[30] Jun Morimoto, Jun Nakanishi, Gen Endo, G. Cheng, Chris Atkeson, and G. Zeglin. Poincare-map-based reinforcement learning for biped walking. In *ICRA 2005: Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2381–2386, April 2005.

[31] Georgio Moroder. Metropolis, 1984.

[32] Franck Multon, Laure France, Marie-Paule Cani-Gascuel, and Giles Debunne. Computer animation of human walking: a survey. *The Journal of Visualization and Computer Animation*, 10:39–54, 1999.

[33] Shinichiro Nakaoka, Atsushi Nakazawa, Kazuhito Yokoi, Hirohisa Hirukawa, and Katsushi Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. In *Proceedings of 2003 IEEE International Conference on Robotics and Automation*, September 2003.

[34] Michiel Van De Panne. *Control techniques for physically-based animation.* PhD thesis, 1994. Adviser-Zvonke G. Vranesic and Adviser-Eugene L. Fiurne.

[35] J. Platt. A resource allocating network for function interpolation. *Neural Computation*, 3:213–225, 1991.

[36] Jörg Sauer and Elmar Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 153–162, New York, NY, USA, 1998. ACM Press.

[37] Stefan Schaal and Christopher G. Atkeson. Receptive field weighted regression. Technical Report TR-H-209, ATR Human Information Processing Laboratories, Kyoto, Japan, 1997.

[38] Stefan Schaal and Christopher G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

[39] Stefan Schaal, Christpoher G. Atkeson, and Sethu Vijayakumar. Real-time robot learning with locally weighted statistical learning. In *ICRA 2000: Proceedings of the 2000 IEEE Internationl Conference on Robotics and Automation*. IEEE, 2000.

[40] Stefan Schaal, Christpoher G. Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for realtime robot learning. *Artifical Intelligence*, 17(1):49–60, 2002.

[41] Stefan Schaal, Sethu Vijayakumar, and Christopher G. Atkeson. Local dimensionality reduction. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 633–639, Cambridge, MA, USA, 1998. MIT Press.

[42] Stefan Schaal, Sethu Vijayakumar, A. D'Souza, A. Isjpeert, and Jun Nakanishi. Real-time statistical learning for robotics and human augmentation. In *International Symposium on Robotics Research*, November 2001.

[43] Nicol N. Schraudoplh, Jin Yu, and Douglas Aberdeen. Fast online policy gradient learning with SMD gain vector adaptation. In *NIPS 2005: Proceedings of the 2005 conference on Advances in neural information processing systems 18*, 2005.

[44] Naresh K. Sinha. *Control Systems*. Holt, Rinehart and Winston, Inc., New York, 1988.

[45] Adnan Sulejmanpašić and Jovan Popovič. Adaptation of performed ballistic motion. *ACM Transactions on Graphics*, 24(1):165–179, 2005.

[46] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. MIT Press, Cambridge, Massachusetts, 1998.

[47] Russ Tedrake. *Applied Optimal Control for Dynamically Stable Legged Locomotion.* PhD thesis, Massachusetts Institute of Technology, September 2004.

[48] Russ Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *IROS 2004: Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2849–2854. IEEE, September 2004.

[49] Honda Robotics `http://asimo.honda.com`.

[50] Vicon Motion Systems `http://www.vicon.com`.

[51] Marc Toussaint and Sethu Vijayakumar. Learning discontinuities with products-of-sigmoids for switching between local models. In *ICML 05: Proceedings of the 22nd international conference on Machine learning*, pages 904–911, New York, NY, USA, 2005. ACM Press.

[52] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys. In *Proceedsing of Graphics Interface '94*, pages 208–215, May 1994.

[53] Sethu Vijayakumar, Aaron D'Souza, and Stefan Schaal. Incremental online learning. Technical Report EDI-INF-RR-0284, University of Edinburgh, 2005.

[54] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005.

[55] Sethu Vijayakumar and Stefan Schaal. Local dimensionality reduction for locally weighted learning. In *CIRA 97: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, page 220, Washington, DC, USA, 1997. IEEE Computer Society.

[56] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *ICML 00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1079–1086, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[57] Chris Welman. *Inverse Kinematics and Geometric Constraints For Articulated Figure Manipulation.* Master's thesis, Simon Fraser University, September 1993.

[58] Andrew Witkin and Michael Kass. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1988. ACM Press.

[59] David Zeltzer. Knowledge-based animation (abstract only). *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer Graphics and interactive techniques*, 18(1):27–27, 1984.

[60] Victor Brian Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 89–96, New York, NY, USA, 2002. ACM Press.