A Phase-Indexed Tracking Controller for Interactive Physical Simulation of Animated Characters

by

Yeuhi Abe

B.S., University of Washington (2003)
S.M., Massachusetts Institute of Technology (2007)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2012

Certified by .....
Jovan Popović
Associate Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

A Phase-Indexed Tracking Controller for Interactive Physical Simulation of Animated Characters

by

Yeuhi Abe

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this thesis, I describe a method of animating characters using physical simulation. The main advantage of this approach, verses traditional keyframing methods, is that the animated character can react to physical interactions. These reactions can be synthesized in real-time in interactive applications, such as video games, where traditional approaches can only playback pre-recorded sequences.

Physically simulating a character requires a controller, but creating a controller is known to be a challenging task, especially when animation concerns about the style of the motion are taken into consideration. This thesis describes a method of generating a controller automatically and quickly from an input motion. The stylistic aspects of the controller are particularly easy to control, as they are a direct result of the input motion.

In order to generate a controller from an input motion, I address two main challenges. First, the input motion must be rectified (minimally modified) to ensure that it is physically plausible. Second, a feedback strategy must be formulated to generate control forces during the simulation. The motion rectification problem is addressed by formulating a fast trajectory optimization that solves for a reference motion. The reference minimally deviates from the input motion to satisfy physical constraints. The second challenge is addressed by employing a novel phase-indexed controller that uses a combination of local and global feedback strategies to keep the character tracking the reference motion. Beyond tracking just a single reference motion, I also demonstrate how variation to a input motion can be automatically synthesized using the same trajectory optimization method used in the rectification process, and how these variations can be sequenced, using optimal control, to accomplish various goals.

Thesis Supervisor: Jovan Popović
Title: Associate Professor

Acknowledgments

Thanks to my research advisor Jovan Popović for taking me under his wing and challenging me to think big and tackle hard problems. I couldn't have made it this far without his encouragement, optimism and expert mentorship. Thanks to all the members of the MIT computer graphics group. Our camaraderie over the years has made me never once question my decision to come to MIT. A special thanks to Marco da Silva, my colleague and friend, for our many fruitful research discussions and collaborations, and for his help defining my career beyond graduate school. (Here's hoping our collaboration continues in the future as well.) Thanks to my thesis readers, Fredo Durand and Russ Tedrake, for their understanding and guidance during the difficult task of dissertation writing. A special thanks to Karen Liu and Zoran Popović, who introduced me to the field of animation research as an undergraduate and encouraged me to apply to graduate school. Finally, thanks to my friends, family, and girlfriend who have stood by me without question while patiently waited for me to shed the title of student.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last three decades has seen the emergence of a new art form. This art form, know as computer animation, was pioneered in the academic realm by computer scientist and then largely cultivated by commercial movie studios such as Lucas Arts, Pixar, and Disney. The fruits of this collective effort have been astounding, as any audience of a recent Hollywood spectacular can attest. Computer animation allows creative story tellers almost limitless potential for conveying their ideas on the screen.

Although the theoretical bounds of an animated scene are limitless, many animations try to replicate or enhance aspects of the real world. Arguably one of the most impressive achievements of this kind is the production of digital characters realistic enough to blend seamlessly with live actors in a movie scene. These characters can perform dangerous stunts that a human actor could not attempt, or be of an alien form that would be hard to achieve through traditional puppetry or costuming techniques. Unfortunately, creating these animations takes large teams of talented artists and engineers countless hours. Although appropriate for blockbuster movies, this approach is not practical for lower-cost productions or for interactive applications where animation that reacts to user input must be synthesized by a computer on-the-fly. For these purposes, a large body of research has investigated software tools for automatic animation of characters. These tools attempt to create high quality, realistic animations quickly (or even interactively) with little or no input

from an animator.

Many software tools for automatic animation use physical laws as guiding constraints for creating realistic motion. Not surprisingly, the use of physical principles in animation even predates the use of computers. For example, early hand animators at Disney espoused the importance of physics in their "12 Basic Principles of Animation" [33]. Several of these principles, such as "stretch and squash", "timing", "arcs", and "slow in/slow out" can be seen as consequences of Newtons laws of physics on a moving body. It has long been understood that human judgment of realism is linked to the perception of whether a motion is physical. For this reason, physical simulation is used ubiquitously for the animation of passive objects, where physical principles alone govern the motion. For example, physical simulation of flowing fluids, wafting gasses, and blending cloth, has long been a mainstay of computer animation in movies and video games alike.

This thesis deals with the subject of physically simulating characters. For characters, such as humans and animals, physics alone is not enough to create convincing animation automatically. Humans and animals are expected to act with intention and with an array of complex motor skills not exhibited by passive objects. Stylistic aspects of human motion are particularly hard to encode in a computer program. Due to these challenges, animation systems have typically shied away from using physical simulation as a means of generating motion. In interactive video games, characters are usually animated with a library of "canned" motions that are carefully sequenced in response to user input. The motions are either created by artists or they are motion capture of actual human actors, but they must be created ahead of time and cannot be synthesized on the fly. This results in an overly repetitive animation that lacks the ability to react to objects which are physically simulated. This is particularly unfortunate, given the increasing use of physical simulation to animate other, non-sentient objects in a scene.

The challenges involved in physically simulating characters are similar to those involved in controlling robots in the real world. In addition to forward integrating the equations of motion for the characters body, the character must be controlled. A controller must be designed in software that continuously observes the current motion of the simulated

character and specifies the internal muscle forces that the character should apply to it's body to achieve desired motions. The controller must also be able to incorporate knowledge of the surrounding environment, so as to appropriately plan the resulting motion. Most important for animation purposes, the stylistic aspects of the controllers must be easily directable and motions must be believable and lifelike.

Since the design of controllers is far from intuitive, this thesis explores an automatic method for taking a traditional "canned" motion sequence as input, and using that as the basis for designing a controller. The advantage of this approach is that it allows a traditional animator to design controllers using tools they are familiar with. Specifying the stylistic aspects of a controller becomes trivial. The controlled character will mimic the style of the input sequence closely until some physical interaction occurs. When this happens, the controller will deviate from the input motion in a physically valid manner, creating interesting and appropriate variation on the original motion.

Developing the tools and methods for automatic synthesis of a controller that can mimic, or track, an input sequence is the main topic of this thesis. The various chapters of this thesis can be seen as contributing toward a unified system for this purpose. There are many facets to this problem. First, it is important to ensure that the input motions that are used to generate controllers are physically feasible in a strict mathematical sense. Otherwise, a physical controller for the motion is impossible. Second, a control strategy must be devised that is capable determining the muscle forces needed to generate the motion as well as robustly handle physical disturbances that cause perturbation in the motion. Lastly, for controllers to be truly useful in an interactive setting, they must be able to switch between different motion behaviors in an intelligent manner in response to the simulated environment or user interaction.

The first of these challenges, that of generating feasible motions, is more troublesome than it might first appear. Animator can create any motion imaginable, but a physical simulation is restricted to those motions that can be produced as a result of forward integration of physical equations of motion. Even if a animator aims to produce motions that look physically plausible, small or imperceptible physical inconsistencies are often enough to cause

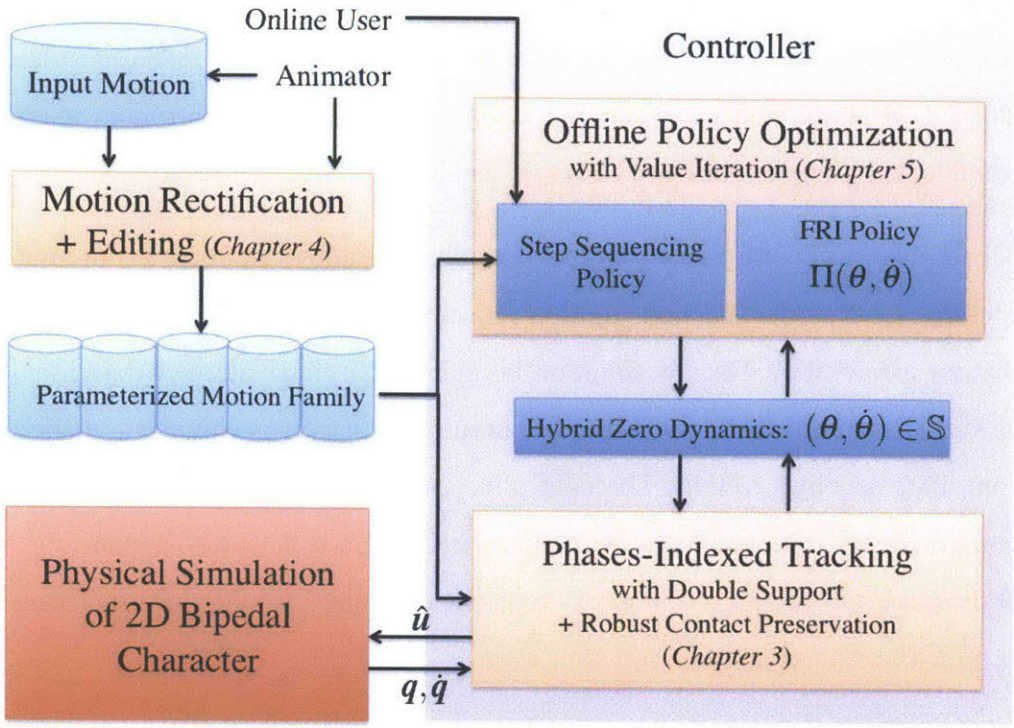

Figure 1-1: *An overview of the complete system described by this thesis. The system allows for the automatic design of stylized controllers using only a single motion as input. The controllers can then be used in conjunction with physical simulation to create interactive character animations.*

trouble in simulation. To address this problem, this thesis discuss the use of trajectory optimization to minimally modify motions so as to ensure they are physically valid (Chapter 4). This is presented as a semi-automatic preprocess that an animator could use in order to ensure physical validity of a motion prior to automatic synthesis of a tracking controller.

However, even when a physically feasible motion is available determining the correct muscle forces to produce the motion is a nontrivial task. A main contribution of this thesis is a method of addressing this challenge through construction of a carefully considered feedback strategy. One of the core components of this strategy is what we call a phase-indexed tracking controller (Chapter 3). The novel aspect of the phase-indexed tracking controller is that it eschews the idea of strictly adhering to a specific motion timing. The input motion is reproduced in terms of overall shape to capture the look and feel of the original, but the

timing is allows to vary in order to ensure stability and robustness in response to perturbation. The advantage of this approach is two-fold. First, since the timing of the motion is allowed to vary, the resulting feedback strategy can be shown to be more robust to other time-indexed motion tracking strategies that lack flexibility in timing. Second, the form of the controller results in a low-dimensional analog of the full system, called a zero dynamics, in which it is possible to make useful predictions about the future state of character. We discuss several different ways in which these predictions can be used to enhance the performance of the controller. In particular, we should how the low-dimensionality of the zero dynamics enables use of an optimal control approach called value iteration (Chapter 5). Value iteration is used to improve the phase-indexed tracking of complex anthropomorphic gait patterns by carefully coordinating forces on the character's feet. Value iteration is also used to help better sequence controllers to navigate a constrained environment or respond to user input.

A main goal of this thesis is to outline a working pipeline for animators to create physically simulated characters. However, many of the methods presented have implications beyond animation. Tracking controllers for artificial legged creatures have implication in fields such as robotics and biomechanics. The ability to accurately reproduce biomimetic motion in simulation may well prove to be a useful tool outside of animation.

Chapter 2

Background

This chapter will provide the reader with the essential background material needed to approach the topics discussed in the remainder of the thesis. The material includes a discussion of methods used to model and simulate a human character, an introduction to controllers at large, an derivation of the basic phase-indexed controller approach that we build upon, and some preliminaries on optimal control methods that are used.

2.1 Physics of Simulated Characters

2.1.1 Equations of Motion

Although the actual physical structure of a human body is vastly complex, a good approximation of the dynamics can be achieved by modeling the various segments of the body as rigid bodies connected by angular joints. Such structures are know as articulated rigid bodies. Their configuration can be represented in a reduced coordinate form [4], in terms of internal joint angles and global orientation and position. Joint angles are given as offsets between parent and child segments in a branching tree structure rooted at an arbitrary segment, at which the root orientation and position is given.The full configuration of the body can be represented by a vector of numbers $\boldsymbol{q} \in \mathcal{Q}$. Instead of modeling actual muscles,

we directly consider torques that act on the angular joints. These forces are represented by a generalized force vector $\boldsymbol{u} \in \mathcal{U}$. We can write the equations of motion in a standard form that explicitly delineates the unactuated root coordinates $\boldsymbol{q}_r$ (and forces $\boldsymbol{u}_r$) from the actuated internal degrees of freedom $\boldsymbol{q}_a$ (and forces $\boldsymbol{u}_a$):

$$\underbrace{\begin{bmatrix} \boldsymbol{M}_r & \boldsymbol{M}_{ra} \\ \boldsymbol{M}_{ra}^\top & \boldsymbol{M}_a \end{bmatrix}}_{\boldsymbol{M}(\boldsymbol{q})} \underbrace{\begin{bmatrix} \ddot{\boldsymbol{q}}_r \\ \ddot{\boldsymbol{q}}_a \end{bmatrix}}_{\ddot{\boldsymbol{q}}} + \underbrace{\begin{bmatrix} \boldsymbol{b}_r \\ \boldsymbol{b}_a \end{bmatrix}}_{\boldsymbol{b}(\boldsymbol{q},\dot{\boldsymbol{q}})} = \underbrace{\begin{bmatrix} \boldsymbol{u}_r \\ \boldsymbol{u}_a \end{bmatrix}}_{\boldsymbol{u}}, \tag{2.1}$$

where $\boldsymbol{M}$ denotes the symmetric, positive definite mass matrix and $\boldsymbol{b}$ is a vector of bias terms.

For the purpose of exposition, let also define both an inverse dynamics function,

$$\boldsymbol{u} = \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}), \tag{2.2}$$

a forward dynamics function,

$$\ddot{\boldsymbol{q}} = \boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}), \tag{2.3}$$

and partial derivatives, $\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{q}}, \frac{\partial \boldsymbol{H}}{\partial \dot{\boldsymbol{q}}}, \frac{\partial \boldsymbol{H}}{\partial \ddot{\boldsymbol{q}}}$.

Note that Equation 2.3 is simply a more general form of Equation 2.1. The specific form of 2.1, simply reflects the fact that character dynamics obey classical Lagrangian mechanics. It is possible to derive this form by writing down the Lagrangian of the system and taking partial derivatives (i.e., calculating the "forced" Euler-Lagrangian equations of motions).

2.1.2 Frictional Contact Dynamics

The equations of motion only describe the effect of internal forces acting on an unconstrained body in free space. The motion of a body in contact with the environment is

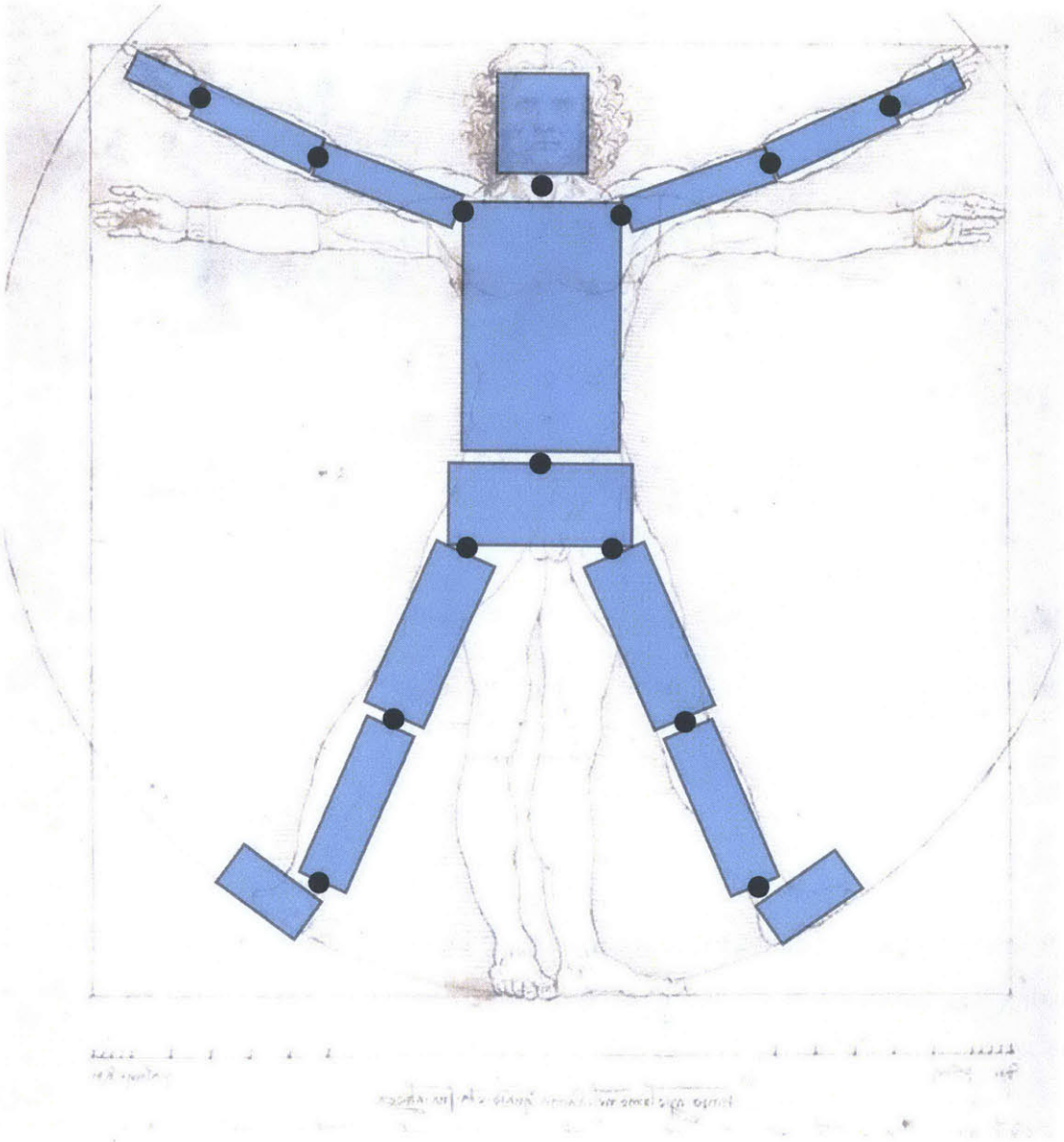

Figure 2-1: *For full body animation, the physical structure of the human body is approximated by rigid bodies (blue boxes) connected by angular joints (black dots).*

significantly more complex due to the presence of contact forces that push on the body to prevent interpenetration between the character and the environment. Although in the real world, contact forces would be the result of many microscopic interactions over the contacting interface between objects, for simulation of characters approximated by idealized rigid bodies, it is sufficient to consider only the interaction of a small number of contact points, $\boldsymbol{p}^{(i)} \in \mathbb{R}^3$ $(i = 1 \ldots m)$ (and forces $\boldsymbol{f}^{(i)} \in \mathbb{R}^3$), that minimally describe the convex hull of the contact interface (see Figure 2-2).

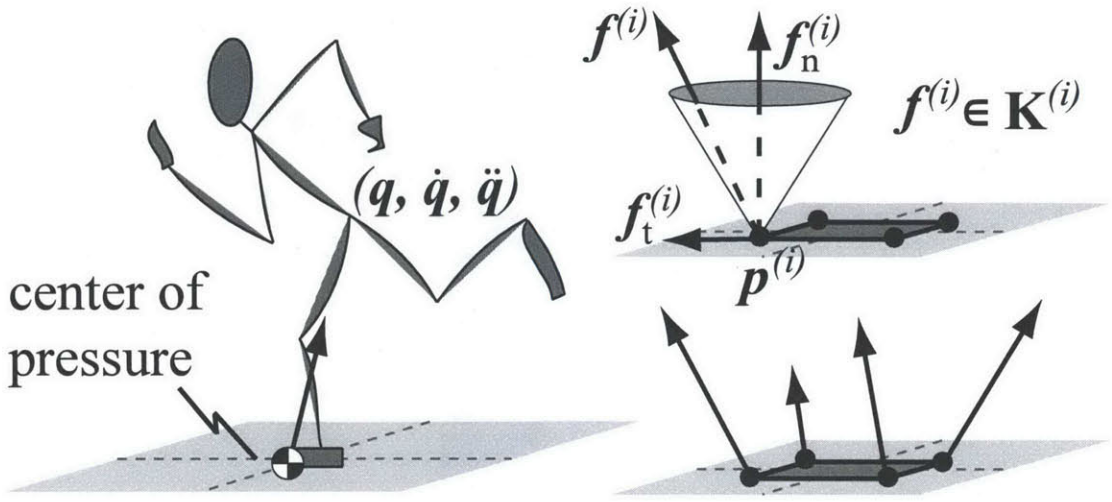

Figure 2-2: *Contact dynamics expresses the relationship between the motion* $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ *of an articulated body, its internal torques, and external forces. We model the contact between two surfaces with a set of point contacts* $\mathbf{p}^{(1)} \ldots \mathbf{p}^{(m)}$ *and the matching contact forces* $\mathbf{f}^{(1)} \ldots \mathbf{f}^{(m)}$. *Each contact force is restricted by a convex cone* $\mathrm{K}^{(i)}$ *according to the well established Coulomb model of friction.*

An analytic model of contact forces can be most intuitively derived by considering a set of unilateral constraints on the acceleration of contact points [3, 55],

$$\ddot{p}_n^{(i)} \geq 0, f_n^{(i)} \geq 0, f_n^{(i)} \ddot{p}_n^{(i)} = 0, \tag{2.4}$$

where the subscript n denotes the component of a vector normal to a contact surface. Starting from any valid, non-interpenetrating contact state with $\dot{p}_n^{(i)} = 0$, the first constraint prevents interpenetration of the contact surface by allowing only positive accelerations in the normal direction. The second constraint prevents the application of forces that pull on the contact point. The third constraint prevents forces from doing work upon the characters. This constraint is often called the complimentary condition since it prevents both $\ddot{p}_n^{(i)}$ and $f_n^{(i)}$ from being nonzero simultaneously.

Additional constraint can be applied to tangential components of the contact forces to en-

force a Coulomb model of friction:

$$\|\boldsymbol{f}_{\mathrm{t}}^{(i)}\| \leq \mu f_{\mathrm{n}}^{(i)}, \boldsymbol{f}_{\mathrm{t}}^{(i)} = \begin{cases} -\mu f_{\mathrm{n}}^{(i)} \frac{\dot{\boldsymbol{p}}_t^{(i)}}{\|\dot{\boldsymbol{p}}_t^{(i)}\|} & \text{if} \|\dot{\boldsymbol{p}}_t^{(i)}\| \neq 0 \\ 0 & \text{if} \|\dot{\boldsymbol{p}}_t^{(i)}\| = 0 \end{cases} \tag{2.5}$$

where the subscript t denotes the two dimensional vector of tangential components and $\mu > 0$ is a coefficient of friction at the contact point. The first constraint ensures that contact forces remain within a friction cone while the second constraint allows only dissipative forces that act counter to the tangential motion along the surface. Together, the constraints on tangential and normal force components characterize a simple but sufficient description of frictional contact dynamics.

The constraints provided here only describe necessary conditions for physical validity. They do not prescribe a method of solving for the contact forces . To solve for the contact forces it is necessary to first observe that there exists a coupling between joint torques and contact forces. Cartesian contact forces acting on the character may be equivalently represented in terms of generalized torques $\boldsymbol{u}$ by transforming them through a well known linear relationship [16, 65],

$$\boldsymbol{u}^{(i)} = [\boldsymbol{G}^{(i)}]^T \boldsymbol{f}^{(i)} \in \mathcal{U}, \tag{2.6}$$

where $\boldsymbol{G}^{(i)} = \frac{\partial \boldsymbol{p}^{(i)}}{\partial \boldsymbol{q}}$ is known as the Jacobian matrix of the ith contact point. Thus the total generalized torque acting on the character,

$$\hat{\boldsymbol{u}} = \boldsymbol{u} + \sum_i \boldsymbol{u}^{(i)}, \tag{2.7}$$

is the sum of both internal muscle forces, $\boldsymbol{u}$, and external contact forces, $\boldsymbol{u}^{(i)}$, coupled through their Jacobian transpose.

This leads to a linear relationship between contact accelerations, $\ddot{p}^{(i)}$, and contact forces, $f^{(j)}$:

$$\begin{aligned}\ddot{p}^{(i)} &= \boldsymbol{G}^{(i)}\ddot{q} + \dot{\boldsymbol{G}}^{(i)}\dot{q} \\ &= \boldsymbol{G}^{(i)}\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{u} + \sum_j \boldsymbol{u}^{(j)}) + \dot{\boldsymbol{G}}^{(i)}\dot{q} \\ &= \boldsymbol{G}^{(i)}\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{u} + \sum_j [\boldsymbol{G}^{(j)}]^T f^{(j)}) + \dot{\boldsymbol{G}}^{(i)}\dot{q}. \end{aligned} \tag{2.8}$$

The relationship is linear because the exact form of the forward dynamics equation $\boldsymbol{F}$ (i.e., solving Equation 2.1 for $\ddot{\boldsymbol{q}}$) is linear in $\boldsymbol{u}$. It can be more compactly represented as,

$$\ddot{p} = \boldsymbol{A}f + \boldsymbol{b}, \tag{2.9}$$

where $\ddot{p} = (\ddot{p}^{(0)}, \cdots, \ddot{p}^{(m)})$ and $f = (f^{(0)}, \cdots, f^{(m)})$ are vector concatenations of all the contact accelerations and forces, and $\boldsymbol{A}$ and $\boldsymbol{b}$ are both directly computable by substituting the exact form of $\boldsymbol{F}$ into the equation above. This linear equation, along with the complimentary constraints (Equation 2.4 and 2.5) form a *complimentary problem* [19] which must be solved to determine contact forces.

2.1.3 Computational Considerations

This section will discuss two important computational choices in the simulation and control of characters in this thesis. The first of these choices is the use of efficient algorithms for the computation of the dynamics functions. These function are used ubiquitously throughout the thesis to describe forward simulation, online control, and offline optimal control algorithms. It is to be assumed that the efficient computational approach described below is used unless noted elsewhere. The second choice discussed is the use of a stable and efficient velocity-based time-stepping scheme for forward simulation, rather than the classic acceleration-based approach.

Efficient Computation of Forward and Inverse Dynamics Functions

When simulating and controlling characters, the forward, $\boldsymbol{F}$ (and often inverse $\boldsymbol{H}$) dynamics function must be computed at each time step, thus for interactive simulation, efficiency becomes an important consideration. Although the form of equation 2.1 provides insight into the structure of the dynamics, in practice it is wasteful to calculate all the coefficients of the $\boldsymbol{M}$ matrix in order to compute the inverse dynamics function. Similarly, despite the fact that the mass matrix $\boldsymbol{M}$ is known to be invertible, performing this inversion to compute the forward dynamics function is not nearly as efficient as directly evaluating the function $\boldsymbol{F}$ through other means.

Featherstone provides, in his book [27], a canonical reference on a family of efficient, recursive algorithms for computing both forward and inverse dynamics for articulated bodies. These methods exploit the tree structure of an articulated rigid body to compute both the inverse, $\boldsymbol{H}$, and forward, $\boldsymbol{F}$, dynamics functions in $O(n)$ time and space, where n is the number of degrees of freedom in the system. They operate by recursively propagating velocities, accelerations, torques, forces, and inertia tensors up and down the tree structure of an articulated body using a convenient "spatial" algebra.

Fang and Pollard [26] have shown that by directly differentiating the recursive Newton-Euler algorithm (described by Featherstone) it is possible to efficiently compute partial derivatives of $\boldsymbol{H}$. Further more, if desired, it is possible to differentiate only a partial subset of the dimensions of $\boldsymbol{H}$, decreasing the number of operations to a fraction of the full cost. Finally, if necessary it is also possible to compute a generalized inertia vector, $\boldsymbol{L} = \boldsymbol{M}\dot{\boldsymbol{q}}$, and its partial derivatives, $\boldsymbol{L}_{\boldsymbol{q}}$ and $\boldsymbol{L}_{\dot{\boldsymbol{q}}}$, using already available byproducts of these other algorithms. If only the generalized inertia terms are needed, a stripped down version of the recursive algorithm is straightforward to formulate. It is even simpler to implement and is faster to compute by an appreciable constant factor.

Acceleration-based vs. Velocity-based Numerical Simulation

Numerical simulation is the process of approximating the solution to a continuous differential equation via a discrete time step. In order to perform numerical simulation in an efficient and stable fashion it is important to consider different numerical approaches. Two possible approaches to simulation are acceleration-based time stepping and impulse-based time stepping. Acceleration-based time stepping is perhaps the most intuitive approach. In this paradigm, the state of the character is forward integrated by formulating a double integrator system,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \tag{2.10}$$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \hat{\boldsymbol{u}}) \end{bmatrix} \tag{2.11}$$

Each simulation step consists of two calculations. First, contacts forces, $f^{(i)}$, are computed by solving the complementarity problem (2.4 and 2.5). Then $\hat{\boldsymbol{u}} = \boldsymbol{u} + \sum_i [\boldsymbol{G}^{(i)}]^T f^{(i)}$ is inserted in the system above and the system is integrated.

However, one issue with this approach is that the contact constraints are only enforced at the acceleration level. Integration error will inevitable causes the velocity state of the constraints to drift from zero. The typical solution to this problem is to use a stabilization method (e.g., Baumgarte stabilization [5]) to correct for deviations from the constraint surface. These methods usually apply a spring-damper-like corrective force to the constraints. However,the coefficients of the spring-damper are typically hard to tune and often inject undesirable (and unrealistic) energy into the system [15].

Alternatively, a velocity-based time stepping approach may be formulated [43, 55, 15], which suggests a different method of handling the contact constraints with some practical advantages. The velocity-based time step is formulated by making two discretization assumptions. First, that the acceleration $\ddot{\boldsymbol{q}}_j$ at time step j may be approximated by a finite

difference,

$$\ddot{\boldsymbol{q}}_j \approx (\dot{\boldsymbol{q}}_{j+1} - \dot{\boldsymbol{q}}_j)/\Delta t, \tag{2.12}$$

and, second, that the contact forces, $f^{(i)}$, may be replaced with contact impulses, $r^{(i)}$:

$$r^{(i)} = \int_t^{t+\Delta t} f^{(i)} \approx \Delta t f^{(i)}. \tag{2.13}$$

Under these discretization assumptions, the equations of motion (2.1) may be rearranged to solve directly for velocities at the next time step:

$$\dot{\boldsymbol{q}}_{j+1} = \dot{\boldsymbol{q}}_j + \boldsymbol{M}^{-1}(\Delta t(\boldsymbol{u} - \boldsymbol{b}) + \sum_i [\boldsymbol{G}^{(i)}]^T r^{(i)}). \tag{2.14}$$

To find the contact impulses, the acceleration-level complimentary contact constraints are reformulated at the velocity-level:

$$[\dot{p}_n^{(i)}]_{j+1} \geq 0, [f_n^{(i)}]_j \geq 0, [f_n^{(i)}]_j [\dot{p}_n^{(i)}]_{j+1} = 0, \tag{2.15}$$

$$\dot{p}_{j+1}^{(i)} = \boldsymbol{G}^{(i)} \dot{\boldsymbol{q}}_{j+1}. \tag{2.16}$$

It can be shown that the acceleration-level constraints (2.5) imply the velocity-level constraints (and visa-versa) by integrating the acceleration constraints (or, alternatively, taking a derivative of the velocity-level constraints).

Rather than solving directly for the contact impulses, they are solved implicitly by find $\dot{\boldsymbol{q}}_{j+1}$ that simultaneously satisfies (2.14), (2.15) and (2.16). This is a complementary problem that maybe solved efficiently using a fast iterative method such as the one described by Erleben [24]. Note that this essentially folds the solution of the contact constraints into the time step. Once the velocity at the next time step are known, the position state can be updated using a implicit Euler integration step: $\boldsymbol{q}_{j+1} = \boldsymbol{q}_j + \Delta t \dot{\boldsymbol{q}}_{j+1}$.

Since the contact constraints are express directly in terms of velocities at the next time step, the constraint velocities are always precise and do not suffer from numerical drift due to

integration error. Due to the dependence of the contact impulses on the velocities at the next time step this is a semi-implicit form of integration. The implicit nature of the time step allows for larger time steps to be taken without numerical instability issues [15]. It has also been noted that when friction terms are included in the contact constraints, this velocity-level formulation of the complementary problem can be shown to always yield a solution where, in the acceleration-level version, it is possible to define physically valid contact configuration for which no there is no solution [55]. Due to these advantages, the velocity-based approach is used for all forward simulation performed in this thesis.

2.2 Control

So far the equations which govern the motion of characters under the constraints imposed by frictional contact have been described. However, the method for determining internal joint torques, $\boldsymbol{u}$, which will cause the character to move in a desired fashion have not been discussed. In the broadest sense, this is the role of control.

This section will start by describing the role of a controller, which is the algorithm that acts online during simulation to determine the correct torques to apply to the character. Next, a description and derivation of a phase-indexed tracking controller will given. This is the basic type of controller upon which this thesis builds. The section will end by first giving a unified view of the concept of optimal control before delving into the preliminaries of two different kinds of optimal control that are used in Chapters 4 and 5.

2.2.1 Controllers

Controllers act in concert with the simulation. At each simulation step, the controller evaluates the current state, $(\boldsymbol{q},\dot{\boldsymbol{q}})^T = \boldsymbol{x} \in \mathcal{TQ}$ of the character in order to determine a control action $\boldsymbol{u} \in \mathcal{U}$ to execute. In most cases, controllers are deterministic in the sense that they correspond to a policy $\Pi(\boldsymbol{x},\boldsymbol{u}) : \mathcal{TQ} \rightarrow \mathcal{U}$, which maps states to control actions. Determining this mapping is the main challenge in designing a controller.

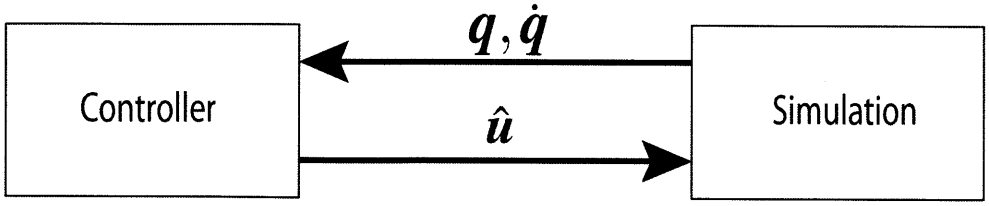

Figure 2-3: *A controller sits in a feedback loop with the simulation. At each simulation step the controller observes the current state* $(\mathbf{q}, \dot{\mathbf{q}})$ *of the character model and determines appropriate internal joint torques* $\mathbf{u}$.

Some of the earliest and most versatile controllers in animation used hand-crafted policies [47, 37, 48, 25, 67] which rely upon clever insights into the character dynamics. Although this approach has seen great success in the research community, it has been difficult to develop tools usable by non-experts. Unlike the approach taken in this thesis, these controllers do not mimic a specific input motion, so it is difficult to control the look or style of the motion. Some [61, 63] have proposed automatic parameter optimization approaches to help in the design process, but these techniques suffer from long search times and tend to require carefully tuned objective shaping terms.

The approach taken in this thesis, of designing controllers that track an input motion, has been previously explored by others in graphics [68, 1, 41, 20, 45]. These previous controllers are best described as time-indexed tracking controllers because they try to mimic the precise timing of the input motion. By contrast, this thesis explores the idea of a phase-indexed tracking controller. The phase-indexed tracking controller reparameterizes a time-indexed reference trajectory in terms of a phase variable. This results in a tracking controller that is time invariant, which has advantages in terms of robustness and versatility. Others have explored similar ideas along the lines of time-invariant tracking. It has been used in the context of autonomous helicopter flight [36] as well as for control of humanoid motion [23, 50].

Most relevant to this thesis is the idea of reparameterizing a time-indexed trajectory in terms of a phase variable. Westervelt and colleagues [64] have shown how this idea can lead to the development of a *hybrid zero dynamics* that can be used to simplify the analysis of biped

motion. Using their approach, they have developed provably stable gait controllers that have been successfully applied to a 7 degree of freedom robot. This thesis develops upon these ideas, applying them to even higher degree of freedom bipeds with more complex gait patterns. However, creation of a hybrid zero dynamics is only one possible use of a phase variable. Manchester and colleagues have shown how a phase variable can also be used to construct a transverse linearization that is used to develop a controller that stabilizes the dynamics to a nominal trajectory using receding horizon control [42]. This idea has been successfully implemented on a 2D compass gait robot. In contrast to the hybrid zero dynamics approach, the transverse linearization approach produces a MIMO linear system for which feedback gains can be computed optimally and automatically.

2.2.2 Phase-Indexed Tracking Controller

In this section the mathematical preliminaries of the phase-indexed controller will be derived in a general form. We will show how, through the use of a specific kind of feedback (which we call *motion constraints*), the dynamics of a high-dimensional system can be effectively reduced to a low dimensional one. The derivation given below is close to the one given by Westervelt and colleagues [64], but similar derivations have been done by others as well (e.g. [51]). The closely related and more general concept of a *zero dynamics* has been know to the nonlinear controls community for some time longer [31, 46].

Motion Constraints

The basic mechanism upon which the phase-indexed controller is constructed is the motion constraint. These constraints are also known as "virtual constraints" in the robotics and control literature. They are often used in the analysis of nonlinear and underactuated control systems [10, 51] because they effectively reduce complex systems to simpler ones that are easier to study. Virtual constraints have been particularly fruitful in the analysis and design of controllers for simplified biped walkers [64, 11, 51, 42].

A preliminary step in the definition of the motion constraints is to define a differentiable

and invertible coordinate transformation,

$$\boldsymbol{g}^{-1}(\boldsymbol{q}) = [\boldsymbol{h}_u(\boldsymbol{q})^T, \boldsymbol{h}_a(\boldsymbol{q})^T]^T : \mathcal{Q} \rightarrow \mathbb{R}^n, \tag{2.17}$$

which maps joint angles to a partitioned set of controlled, $\boldsymbol{h}_a \in \mathbb{R}^m$, and uncontrolled, $\boldsymbol{h}_u \in \mathbb{R}^{n-m}$, coordinates. The size of $\boldsymbol{h}_a$ is assumed to be the same as the number of actuated degrees of freedom.

In addition to the coordinate transform, a smooth differentiable scalar function

$$\theta(\boldsymbol{h}_u) \tag{2.18}$$

is defined. For motion tracking purposes, θ is defined such that its expected evolutions in time $\theta(t)$ is smooth and monotonic. This is because θ is to serve as a replacement for the time variable in the phase-indexed tracking controller and we must ensure that a smooth one-to-one mapping exists between θ and time for the type of motion the controller is to perform. Due to this restriction, θ will be referred to as the monotonic phase variable.

The motion constraints $\boldsymbol{y}$ are now given by a relationship of the form

$$\boldsymbol{y}(\boldsymbol{q}) = \boldsymbol{h}_a(\boldsymbol{q}) - \boldsymbol{c}(\theta) = 0, \tag{2.19}$$

where $\boldsymbol{c}(\theta)$ is a differentiable function that is chosen as part of the controller design process.

Since the number of motion constraints $\boldsymbol{y}$ is the same as the number of actuated degrees of freedom, precise control is possible through a *partial feedback linearization* [53] of the form:

$$\begin{bmatrix} \ddot{\boldsymbol{h}}_u \\ \ddot{\boldsymbol{y}} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{A}^0_{h_u} & \boldsymbol{A}^1_{h_u} \\ \boldsymbol{A}^0_{y} & \boldsymbol{A}^1_{y} \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} \boldsymbol{u}_a \\ \boldsymbol{u}_r \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{d}_{h_u} \\ \boldsymbol{d}_{y} \end{bmatrix}}_{\boldsymbol{d}}, \tag{2.20}$$

where $\boldsymbol{A}$ and $\boldsymbol{d}$ are easily computed by twice differentiating $\boldsymbol{y}$ and $\boldsymbol{h}_u$ with respect to q and plugging in the forward dynamics equation (2.3). The assumption that the root degrees of

freedom, $\boldsymbol{q}_r$, are unactuated implies that $\boldsymbol{u}_r = 0$. This prevents direct control of both $\ddot{\boldsymbol{h}}_u$ and $\ddot{\boldsymbol{y}}$ simultaneously, but the motion constraints can be stabilized around $\boldsymbol{y} = 0$ by applying feedback,

$$\boldsymbol{u}_a = (\boldsymbol{A}_{\boldsymbol{y}}^0)^{-1}(\ddot{\boldsymbol{y}}^* - \boldsymbol{A}_{\boldsymbol{y}}^1 \boldsymbol{u}_r - \beta_{\boldsymbol{y}}), \tag{2.21}$$

$$\ddot{\boldsymbol{y}}^* = -\frac{1}{\varepsilon} k_s \boldsymbol{y} - \frac{1}{\varepsilon^2} k_d \dot{\boldsymbol{y}}, \tag{2.22}$$

where k_s and k_d are gains and ε controls the exponential rate of convergence of $(\boldsymbol{y}, \dot{\boldsymbol{y}})$ to $(0,0)$.

Reduced System

One of the main advantages of using motion constraints is that they result in a new system dynamics with reduced dimensionality. When the motion constraints are enforced (i.e., $\boldsymbol{y} = 0$), the state can be expressed solely in terms of the uncontrolled coordinates $\boldsymbol{h}_u$:

$$\boldsymbol{q} = \boldsymbol{g}(\boldsymbol{h}_a, \boldsymbol{h}_u) = \boldsymbol{g}(\boldsymbol{c}(\theta(\boldsymbol{h}_u)), \boldsymbol{h}_u) \triangleq \boldsymbol{q}_{\boldsymbol{c}}(\boldsymbol{h}_u), \tag{2.23}$$

$$\dot{\boldsymbol{q}} = \frac{\partial \boldsymbol{q}_{\boldsymbol{c}}}{\partial \boldsymbol{h}_u} \dot{\boldsymbol{h}}_u, \tag{2.24}$$

where $\boldsymbol{q}_{\boldsymbol{c}}(\boldsymbol{h}_u)$ is a computable function that reconstructs the full state from only the uncontrolled coordinates. The subscript $\boldsymbol{c}$ denotes the dependence upon the motion constraint function. Note that this reconstruction is only valid when $\boldsymbol{y} = 0$ and, similarly, the relationship (2.24) is only valid when $(\boldsymbol{y}, \dot{\boldsymbol{y}}) = (0,0)$.

The subset of states that can be represented by $\boldsymbol{q}_{\boldsymbol{c}}(\boldsymbol{h}_u)$ (and its tangent space):

$$\{(\boldsymbol{q}, \dot{\boldsymbol{q}}) \| (\boldsymbol{y}, \dot{\boldsymbol{y}}) = 0\} \subset \mathcal{TQ} \ni (\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.25}$$

is called the zero dynamics set.

The resulting closed-loop dynamics of the partial feedback linearization (2.20),

$$\begin{aligned} \ddot{\boldsymbol{h}}_u &= \boldsymbol{f}(\boldsymbol{h}_u, \dot{\boldsymbol{h}}_u, \boldsymbol{h}_a, \dot{\boldsymbol{h}}_a) + \boldsymbol{g}(\boldsymbol{h}_u, \dot{\boldsymbol{h}}_u, \boldsymbol{h}_a, \dot{\boldsymbol{h}}_a)\boldsymbol{u}_r \\ \ddot{\boldsymbol{y}} &= \ddot{\boldsymbol{y}}^*(\boldsymbol{h}_u, \dot{\boldsymbol{h}}_u, \boldsymbol{h}_a, \dot{\boldsymbol{h}}_a), \end{aligned} \tag{2.26}$$

may be derived by substitution of (2.21) into (2.20).

However, when the constraints have stabilized to the set $(\boldsymbol{y}, \dot{\boldsymbol{y}}) = (0, 0)$, we have that

$$\boldsymbol{h}_a = \boldsymbol{c}(\theta(\boldsymbol{h}_u)) \tag{2.27}$$

$$\dot{\boldsymbol{h}}_a = \frac{\partial \boldsymbol{c}(\theta(\boldsymbol{h}_u))}{\partial \theta} \frac{\partial \theta(\boldsymbol{h}_u)}{\partial \boldsymbol{h}_u} \dot{\boldsymbol{h}}_u \tag{2.28}$$

and by substituting (2.27) and (2.28) into (2.26) we find that the dynamics may be written in a much reduced form,

$$\begin{aligned} \ddot{\boldsymbol{h}}_u &= \boldsymbol{f}(\boldsymbol{h}_u, \dot{\boldsymbol{h}}_u) + \boldsymbol{g}(\boldsymbol{h}_u, \dot{\boldsymbol{h}}_u)\boldsymbol{u}_r \\ \ddot{\boldsymbol{y}} &= 0, \end{aligned} \tag{2.29}$$

which only has dependence upon the uncontrolled coordinate state $(\boldsymbol{h}_u, \dot{\boldsymbol{h}}_u)$. This system is called the *zero dynamics* of the constraints $\boldsymbol{y}$. Since the number of uncontrolled coordinates is equal to the number of unactuated degrees of freedom, and this is usually only a small fraction of the full dimensionality of the character, this system allows for a low dimensional analysis of the character's behavior under the action of the motion constraints.

2.2.3 Optimal Control

The easiest way to specify a control policy is often to define a cost function that should (ideally) be minimized by the policy. In many cases, this cost function can be represented by a sum (or integral) of an instantaneous costs over time:

$$C = \sum_i c(\boldsymbol{x}_i, \boldsymbol{u}_i, i), \tag{2.30}$$

where the instantaneous cost at each time index i is a function of the state and the control.

For example, a tracking controller might define an instantaneous cost function that penalizes the deviation of the character's state, $\boldsymbol{x}$, from a desired state, $\hat{\boldsymbol{x}}_i$:

$$c(\boldsymbol{x}, \boldsymbol{u}, i) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}_i\|^2. \tag{2.31}$$

However, determining the control policy that minimizes a given cost function for all possible states is often quite difficult, if not impossible. Further compounding this problem is the fact that characters (e.g., bipeds) are underactuated, meaning that they possess less control inputs than degrees of freedom. This deficit of controls may be expressed in terms of differential constraints on the allowable trajectories of the system. In layman's terms, this means that not all trajectories are feasible and, furthermore, it is nontrivial to express which ones are infeasible. An optimal policy must account for this kind of complication. Optimal control methods attempt to solve this problem using a variety of techniques. Due to the complexity and dimensionality of the problem, numerical methods are often employed and the resulting policy, $\Pi(\boldsymbol{x})$, is often only an approximation of a local minima in the global landscape of possible functions.

An important distinction exists between optimal control methods that determine a feedforward policy and those that determine a feedback policy. Feedforward policies only provide a partial mapping of Π, where the domain is restricted to a 1-dimensional subset of the state space along a (locally) optimal trajectory. In fact, the output of these methods is typically just a trajectory in state space along with the corresponding sequence of control actions. The control actions correspond to the trajectory in the sense that if they are executed starting from a specific initial condition, the system dynamics will proceed along the given

trajectory. Thus these feedforward methods are often just called trajectory optimizations.

Feedforward trajectory optimizations do not provide a contingency for when the system's state diverges from the given trajectory. Thus even small disturbances will tend to precipitate into larger deviations, which can quickly lead to failure. Feedback policies provide a larger set of states for which of the optimal (or approximately optimal) policy can be computed. A global feedback policy provides an optimal control action for every possible state. However, the dimension of such a policy will inevitably scale with the dimension of the character. Thus for high-dimensional, non-linear, underactuated systems (as is the case for characters) it is usually impossible to determine a globally optimal feedback policy, except in special cases.

A vast literature on the topic of optimal control deals precisely with this problem. A typical solution is to approximate the optimal policy by reducing the effective dimension of the system state using clever (and often problem dependent) projections or mappings (e.g., [20, 18]). Another trick often employed for nonlinear systems is to compute an approximately optimal policy in some small local vicinity of an optimal trajectory. This is often done by computing a Taylor series approximation of the full nonlinear dynamics along the feedforward trajectory and using this in place of the full dynamics for sufficiently nearby states. By approximating the nonlinear system as a time varying linear system, and by integrating the so called Riccati equations [54, 34], a time varying version of the standard Linear Quadratic Regulator (LQR) [54] may be used to generate linear feedback gains for a nonlinear system. Another almost identical approach, known as differential dynamic programming (DDP) [32, 2], has been explored more recently for robotics control applications. Such methods are better than a feedforward policy, but still lack convergence guarantees away from the trajectory upon which they are developed. This has lead to recent efforts to improve these methods by stitching together many local trajectories in order to define a policy function on a larger subset of the domain [58, 57].

Feedforward trajectory optimization coupled with linear feedback based upon LQR or DDP are usually used together. First, an optimal feedforward trajectory is computed. This trajectory is then used as the local set of states upon which Taylor series expansion of the dynam-

ics and integration of the Riccati equations is performed. Although these approaches show promise, there are many technical details and complications in the practical implementation which have so far inhibited their wide application. The work described in this thesis leverages feedforward trajectory optimization, but it avoids the complications of feedback using LQR or DDP type methods. Instead, a somewhat different approach is developed. Feedback is composed of two parts: a local, non-optimal feedback policy based upon motion constraints and an optimized policy that operates on the resulting zero dynamics. Since the zero dynamics is of a small enough dimension ($n = 2$), we apply a different kind of optimal control, know as *value iteration*, that attempts to determine a global policy function on the zero dynamics. In so far as the full dimensional system approximates the zero dynamic, the global policy will be behave approximately optimally on the full dimensional system as well. Although we do not find strict approximation bounds on the optimality, in practice this approach results in reasonable feedback policies on the full system.

The following two section will give the reader general background on the methods used in this thesis to find optimal feedforward trajectories (direct transcription) and to determine a global policy on the zero dynamics (value iteration). Further development and application of these methods to the specific character control problems considered in this thesis will occur in Chapters 4 and 5.

Direct Transcription Methods

Direct transcription methods are ways of solving optimal feedforward trajectories using optimization. They solve for optimal trajectories by "transcribing" a control problem into a standard nonlinear programming problem (NLP). When using a direct transcription methods, continuous state and control functions are discretized in time. The discrete values become the free variables in an NLP. The objective function of the NLP represents the cost function to be minimized by the optimal trajectory. The constraints of the NLP enforce the physical dynamics of the system as well as other problem specific limits on the state and the control.

As an example of a simple direct transcription problem, let us consider the trajectory $\boldsymbol{x}(t)$ of a particle mass m subject to time varying control forces $\boldsymbol{f}(t)$. The second-order dynamics are given by $m\ddot{\boldsymbol{x}} = \boldsymbol{f}$. If we wish to solve for a minimal force trajectory that transport the particle from point $\boldsymbol{a}$ to point $\boldsymbol{b}$ while avoiding the unit disc located at the origin, we might solve the following NLP:

$$\begin{aligned} \min_{\{\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i, \boldsymbol{f}_i | i=1\ldots n\}} \quad & \textstyle\sum_i \|\boldsymbol{f}_i\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{a} \\ & \boldsymbol{x}_n = \boldsymbol{b} \\ & \dot{\boldsymbol{x}}_i = \tfrac{\boldsymbol{x}_{i+1} - \boldsymbol{x}_i}{\Delta t} \\ & \tfrac{\boldsymbol{f}_i}{m} = \tfrac{\dot{\boldsymbol{x}}_{i+1} - \dot{\boldsymbol{x}}_i}{\Delta t} \\ & (x_i^0)^2 + (x_i^1)^2 > 1 \end{aligned}$$

The free variables in the problem above represent the discrete positions, velocities, and forces (resp. $\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i, \boldsymbol{f}_i$) along the optimal trajectory. The NLP minimizes the discrete forces subject to the constraints. The first two constraints enforce the initial and final position of the particle. The third and fourth constraints enforce a "forward Euler" (or finite difference) relationship between adjacent positions and velocities and adjacent velocities and accelerations. The final constraint prevents the trajectory from intersecting the unit disc. Although the constraints above are formulated using an forward Euler difference equality, it is equally possible to use any higher order or implicit difference formula. The various possible difference methods are also called collocation methods, of which there are many. A comprehensive treatment of the subject can be found in the book by Betts [8].

In computer graphics, direct transcription has been used for motion design and editing and can be seen as a variant of the *spacetime constraints* method first introduced to the field by Witkin [35]. In aeronautics and robotics, direct collocation is a standard tool for trajectory synthesis because it generalizes well to a broad range of problems and because there are many mature numerical codes for solving NLP problems. In this thesis, direction transcription is used to find optimal motions for simulated bipeds. Chapter 4 is devoted to the description of this method.

Value Iteration

Value iteration is an optimal control algorithm that is often able to determine globally optimal policies for discrete problems with a small number of dimensions and a finite number of states. It can also be used to find approximate solutions to continuous problems by discretizing the state and control spaces. Value iteration works by leveraging the existence of optimal substructure inherent to many optimal control problems. Optimal substructure means that the optimal control problem can be divided into smaller subproblems, the solution to which can be used to solve larger subproblems. This process is repeat until the whole solution is know. In this light, value iteration may be seen as a form of dynamic programming.

Value iteration operates on a discrete system with state $s \in \mathbb{S}$, control actions $a \in \mathbb{A}$, and a transition function $\mathbf{T}(s,a) : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S}$, which maps state and action pairs to new states. The goal of value iteration is to find an globally optimal policy, $\Pi^* : \mathbb{S} \rightarrow \mathbb{A}$, which minimizes a given cost function.

Value iteration works under the assumption that an instantaneous cost $c(s,a)$ is given for being in state s and performing action a. The value, $\mathcal{V}(s_0)$, for starting in state s_0 is given in terms of a sum of all instantaneous costs for s_0 and all subsequent states s_i, $i > 0$:

$$\mathcal{V}(s_0) = -\sum_i \alpha^i c(s_i, a_i), (0 \leq i \geq n) \tag{2.32}$$

where the term $\alpha \in [0,1]$ is a discount factor that is often included to express diminishing dependence upon states farther into the future.

The optimal substructure inherent in the problem may be made explicit by rewriting the value function recursively as

$$\mathcal{V}(s_i) = -c(s_i, a_i) + \alpha \mathcal{V}(s_{i+1}). \tag{2.33}$$

Note, however, that computing the value function requires knowledge of future states s_i, but the sequence of future states will depend upon the policy. If the value function is used as a criteria to design the policy, this would seem to lead to circular reasoning. However, Bellman showed that it is often possible to solve for both the value function and the optimal policy efficiently! This is done by observing a criteria of optimality that must be satisfied by the value function and policy.

Even without knowing the value function, we can derive the following relationships that must hold for the optimal policy, Π^*,

$$\Pi^*(s_i) = \arg\max_{a \in \mathbb{A}} [c(s_i, a) + \mathcal{V}(\mathbf{T}(s_i, a))], \tag{2.34}$$

$$\begin{aligned} \mathcal{V}(s_i) &= c(s_i, \Pi^*(s_i)) + \mathcal{V}(\mathbf{T}(s_i, \Pi^*(s_i)) \\ &= \max_{a \in \mathbb{A}} [c(s_i, a) + \mathcal{V}(\mathbf{T}(s_i, a))]. \end{aligned} \tag{2.35}$$

The first equation above shows that once the value function is known, the optimal policy is simply the greedy one which chooses the action that transitions to the state with maximum value. The second equation, known as the Bellman equation, expresses a condition of optimality for the value function. It can be shown (under normally satisfiable conditions) that the value function for the optimal policy uniquely satisfies the Bellman equation.

Value iteration iteratively refines the solution of the value function by alternately updating the two equations above. Given an initial guess at the value function, the policy is updated for each state using equation (2.34). Then the value function is recomputed for each state using equation (2.35). This process is repeated until convergence. There are many strategies for ordering the updates to the value function. Some strategies update states in an asynchronous fashion where the same state is updated multiple times before performing an update to another state. Different strategies many be more advantageous depending upon the structure of the transition function.

A excellent reference on value iteration, with its many variations, can be found in [7]. In

particular, one useful variation of the basic algorithm, as outlined here, is to allow for probabilistic transitions between states. In this case, the meaning of the transition function is modified to be a probability distribution over possible states. The value function is likewise modified to compute the expected value:

$$\mathcal{V}(s_i) = -c(s_i, a_i) + \alpha \sum_j \mathcal{V}(s_j) p(s_j | s_i, a_i), \tag{2.36}$$

, where $p(s_j|s_i, a_i)$ is the probability of transitioning to state s_j when starting in state s_i and performing action a_i. Value iteration can be used to optimize this expected value.

Chapter 3

Phase-Indexed Tracking Controller

This chapter will describe our implementation and experiments with a phase-indexed tracking controller for a simulated 2D biped. It will be shown how the phase-indexed tracking controller can be used to control a simulated character in a manner that captures the overall style of a provided input motion.

At the core of the controller is a specific form of the general motion constraints described in the previous chapter (Section 2.2.2). This specification will involve defining the invertible coordinate transform $\boldsymbol{g}^{-1}(\boldsymbol{q})$, the monotonic phase variable θ, and the constraint functions $\boldsymbol{c}(\theta)$. The derivation of the motion constraints will assume a simplified model of the character with a fixed point foot. However, the motion constraints on the simplified model are only used as starting point for computing the output to the full character. Additional calculations augment the basic motion constraint feedback to account for the presence of feet, double support, and frictional contact constraints. Feet are accounted for by allowing some limited actuation of the (initially assumed unactuated) root degree of freedom of the simple model. This actuation is calculated by defining an auxiliary control policy that places the position of the ground reaction force on the foot. The resulting ankle actuation is coupled back into the feedback on the motion constraints in a principled manner. Double support stages of motion are handled by a slight modification to the motion constraints, which accounts for the closed-loop kinematic configuration of the legs. Finally, constraints

on unilateral and frictional contact dynamics are handled by allowing a graceful degradation of the motion constraints to occur in extreme situations when these constraints would otherwise become violated. This degradation is formulated as a constrained, least square optimization on joint accelerations, which are then mapped back to joint torques as the final output of the controller.

We demonstrate that the resulting controller is able to perform several different stylized forms of walking and even a jumping motion (with only slight modification). The controller is remarkably robust to unexpected disturbances such as uneven ground and external forces applied to the body of character. We also show that the resulting controller is significantly more robust to force disturbances than a state-of-the-art time-indexed tracking controller based upon the Nonlinear Quadratic Regulator (NQR).

3.1 Motion Constraints for 2D Bipeds

We will start by formulating a set of motion constraints for a simplified biped with a fixed point contact between a foot and the ground. That is to say, initially we ignore the existence of the feet and only consider the contact between the ground and one of the two legs at any given moment in time. We also initially ignore the existence of unilateral constraints on the normal contact forces and frictional constraints on tangential contact forces. This is somewhat justified by the fact that during normal modes of operation, the control will not expect to violate these constraints due to the downward pull of gravity. Only for large disturbances to the motion will we expect these constraints to play a significant role in recovery. Once the basic formulation for control with a single point foot without friction has been described, modifications for principled handling of feet, double support, and frictional contact dynamics will be introduced.

Since contact is initially limited to a single point, without restrictions on contact forces, we can consider the dynamics of a simplified character with only single root degree of freedom, $\boldsymbol{q}_r$, which is the angle of the bottom leg segment with respect to the ground (see Figure 3-1).

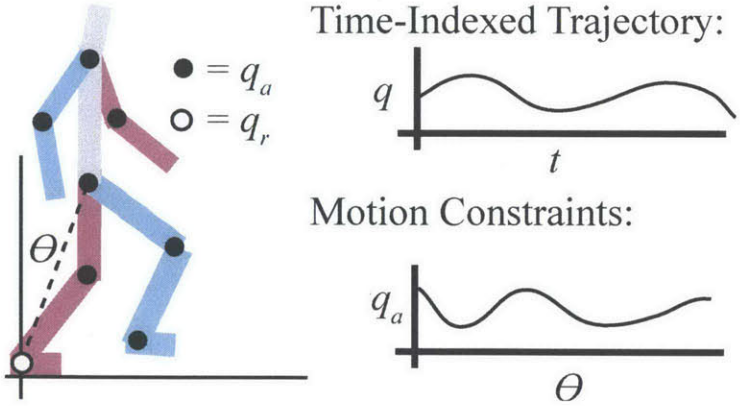

Figure 3-1: *Motion constraints enforce a kinematic relationship between a state variable θ and the actuated degrees of freedom $\mathbf{q}_a$. The parameters of the constraints are fit to match a specific input motion. They serve a similar function to the trajectories tracked by time-indexed controllers, but without enforcing a specific timing.*

With these assumption in place, we define an invertible coordinate transform (2.17)

$$\boldsymbol{g}^{-1}(\boldsymbol{q}) = [\boldsymbol{h}_u(\boldsymbol{q})^T, \boldsymbol{h}_a(\boldsymbol{q})^T]^T \tag{3.1}$$

$$\boldsymbol{h}_u = \theta, \qquad \boldsymbol{h}_a = \boldsymbol{q}_a. \tag{3.2}$$

This particularly simple and effective choice of the coordinate transform was first introduced by Westervelt and colleagues [64].

The choice of the monotonic phase variable θ depends upon the type of motion being tracked by the controller, however, two particularly convenient choices that we have experimented with are the angle between the hip and the point contact with the ground (see Figure 3-1) and the horizontal displacement of the center of mass and ground.

Given the coordinate transform above, the motion constraints can be expressed as

$$\boldsymbol{y}(\boldsymbol{q}) = \boldsymbol{q}_a - \boldsymbol{c}_w(\theta), \tag{3.3}$$

where the constraint functions $\boldsymbol{c}_w$ are defined by piecewise B-spline curves with control points w.

Note that since we have defined $\theta = \boldsymbol{h}_u$ and $\boldsymbol{h}_u$ as a function of $\boldsymbol{q}$, we may for notational

convenience express θ in any of the following equivalent symbolic forms:

$$\theta = \boldsymbol{h}_u = \theta(\boldsymbol{h}_u) = \theta(\boldsymbol{q}) = \theta(\boldsymbol{q}_a, \boldsymbol{q}_r). \tag{3.4}$$

Furthermore, for notational convenience we will allow the reconstruction function $\boldsymbol{q}_c(\boldsymbol{h}_u)$ (2.23) to be written equivalently in any of the following forms:

$$\boldsymbol{q}_c(\boldsymbol{h}_u) = \boldsymbol{q}_c(\theta) = \boldsymbol{q}_w(\theta), \tag{3.5}$$

where the subscript w again expresses the dependence of the reconstruction function on the B-spline parameters of the constraint functions. An important consequence of the above is that when $(\boldsymbol{y}, \dot{\boldsymbol{y}}) = (0,0)$ the full state of the character may be reconstructed from θ and $\dot{\theta}$ alone:

$$\boldsymbol{q} = \boldsymbol{q}_w(\theta) \tag{3.6}$$

$$\dot{\boldsymbol{q}} = \frac{\partial \boldsymbol{q}_w(\theta)}{\partial \theta}\dot{\theta}. \tag{3.7}$$

3.2 Design of Motion Constraints

Motion constraints are designed to mimic the joint configurations of an input motion. This is done by specifying the motion constraint functions $\boldsymbol{c}_w(\theta)$ (defined by parameters $\boldsymbol{w}$) that best fit the motion. In our construction we use piece-wise B-spline curves [21] to represent the $\boldsymbol{c}_w$ functions over the motion. Continuous segments of the curve are given by individual B-spline curves $\boldsymbol{c}_{w_i}$, $w = [w_0 \ldots w_n]$, where n is the number of segments. Thus the parameters $\boldsymbol{w}$ for which we must solve correspond to control points of the B-spline segments.

Given an input motion we wish to track, the process begins by dividing the motion into different stages depending on the contact configuration between the character and the ground.

The simple point contact model used to develop the motion constraints will vary depending upon how the character makes contact with the ground (Figure 3-2. During stages where at least one of the feet is flat, the simple model is rooted at the ankle joint of the flat foot. Since the foot is flush with the ground and is not expected to move during this stage of the motion, we need not consider it's motion. During stages when no foot is flat on the ground, the simple model is rooted at which ever point is making contact with the ground. We must include the foot link in the simplified point foot model during these stages of the motion.

In between some stages of the motion, impulsive collisions change the velocity state of the character discontinuously. This happens, for example, when a foot strikes the ground. Whenever a discontinuity in the velocity state occurs, we transition to a new B-spline segment to better represent this discontinuity. We also transition to a new B-spline segment whenever the topology of the simple point foot model changes (Figure 3-2). However care must be taken to ensure that the end of one B-spline segment is consistent with the beginning of a new B-spline segment. We refer to this as the consistency condition.

To express the consistency condition, we start with the assumption that contacts occur as a standard inelastic impulse between the character and the environment

$$\boldsymbol{L}\dot{\boldsymbol{q}}^{-} = \dot{\boldsymbol{q}}^{+}, \tag{3.8}$$

where $\dot{\boldsymbol{q}}^{-}$ and $\dot{\boldsymbol{q}}^{+}$ are the joint velocities before and after the impulse and $\boldsymbol{L}$ is a linear map that depends only on the configuration of $\boldsymbol{q}$ at the impulse event. When no impulsive contacts occur, $\boldsymbol{L}$ is simply the identity matrix. The form of $\boldsymbol{L}$ is identical to the linear impulse assumption used by Westervelt [64] and can be derived by solving for the contact impulses necessary to bring the velocity of the newly formed contact point immediately to zero.

By substituting (3.7) into (3.8) a consistency condition,

$$0 = \boldsymbol{C}(w_0, w_1) = \boldsymbol{L}\frac{\partial \boldsymbol{q}_{w_0}(\theta_0)}{\partial \theta} - \frac{\partial \boldsymbol{q}_{w_1}(\theta_0)}{\partial \theta}\frac{\partial \theta}{\partial \boldsymbol{q}}\boldsymbol{L}\frac{\partial \boldsymbol{q}_{w_0}}{\partial \theta_0}, \tag{3.9}$$

for adjacent B-spline segments of the constraint curve may be derived, where θ_0 is the value

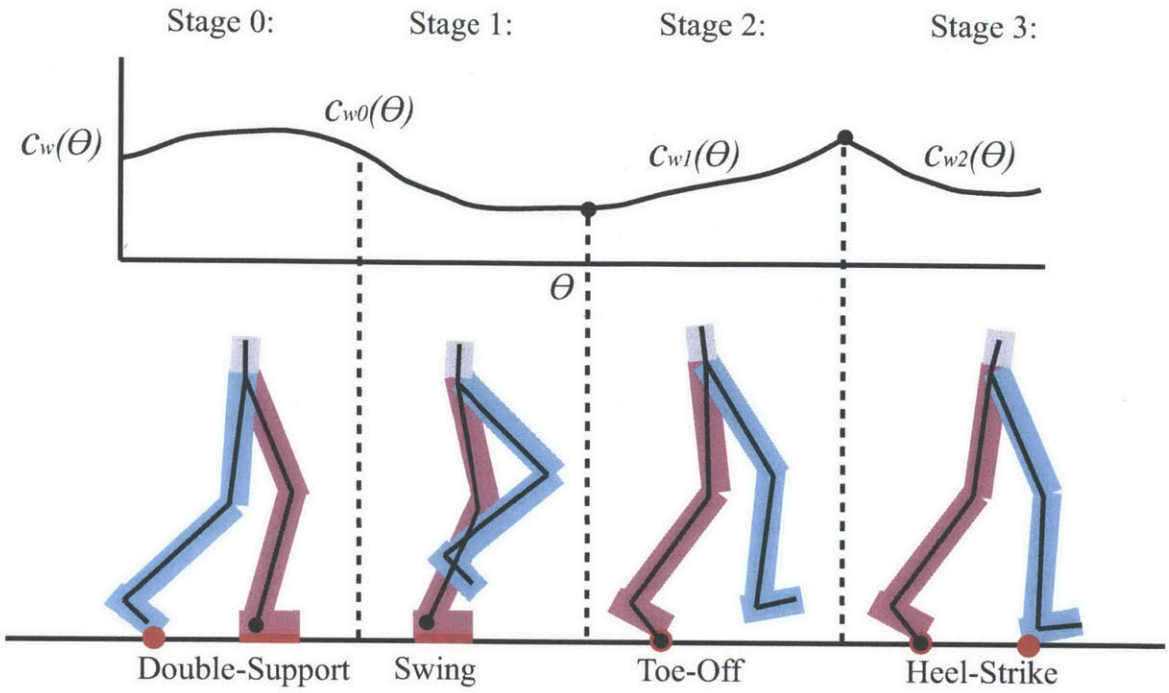

Figure 3-2: *This figure depicts the four different stages of a typical anthropomorphic walking gait: double-support, swing, toe-off, and heel-strike. The red lines and dots depict the contact between the feet and the ground. The black lines depict the topology of the simple, point foot model used during each stage. At the top is a representation of the B-spline constraint function. Transitions between different piecewise segments of the B-spline constraint function occur whenever either the topology of the simple model changes or an impulsive contact collision occurs, such the heel striking the ground between the toe-off and heel-strike stages.*

of θ when the switch between B-spline segments occurs and $\boldsymbol{q}_{w_0}(\theta_0)$ and $\boldsymbol{q}_{w_1}(\theta_0)$ are the motion reconstruction functions (3.6) computed using adjacent B-spline segments $\boldsymbol{c}_{w_0}$ and $\boldsymbol{c}_{w_1}$ evaluated at θ_0.

Finally, we solve for control points $\boldsymbol{w}$ that satisfy the consistency condition and that minimize $\boldsymbol{y}$ along the trajectory of the input motion. The optimization takes the form

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \sum_{i=0}^{n} (\|\hat{\boldsymbol{q}}_a(t_i) - \boldsymbol{c}_w(\hat{\theta}(t_i))\|_2 + k\|\frac{\partial^2 \boldsymbol{c}_w(\hat{\theta}(t_i))}{\partial \theta^2}\|_2) \\ \text{subject to} \quad & \boldsymbol{C}_j(w_j, w_{j+1}) = 0, \end{aligned} \tag{3.10}$$

where n is the number of discrete sample points along the input motion, t_i is the time of

sample point i, $\hat{\boldsymbol{q}}_a(t_i)$ and $\hat{\theta}(t_i)$ are the nominal values of the actuated degrees of freedom and θ at time t_i in the input motion, respectively, and $\boldsymbol{C}_j$ is the consistency condition between B-spline segment j and $j+1$. k is a scalar regularization term that helps to avoid large accelerations in the motion constraints that would make their derivatives poorly conditioned.

The consistency conditions are nonlinear in the parameters $\boldsymbol{w}$ because the impulse map $\boldsymbol{L}$ depends upon the configuration of the character at the transition between stages. So we solve the system using nonlinear optimization with finite differencing of the gradients. We found that solutions could be obtained in less than a minute on a midrange desktop computer, for a 2D character with 14 degrees of freedom for up to 2 seconds of input motion.

Although our method of designing the motion constraints differs from prior work, the resulting point-foot walking controller closely resembles the one described by Westervelt and colleagues [64]. In fact, many of our controllers exhibit the passive stability characteristic of the walking controllers designed by Westervelt, despite the fact that we do not specifically strive to design motion constraints with this property. Instead of passive stability, the focus of our motion constraints design is on accurate reproduction of the configurations of the input motion.

3.3 Robust Contact and Double Support

This section will describe ways in which the basic feedback provided by the motion constraints is augmented to develop a robust controllers that uses the feet advantageously and that can handle the presence of double support.

3.3.1 FRI Policy and Contact Preservation

In the development of the motion constraints thus far, a fixed connection between a point foot and the ground has been assumed. This simple model captures the gross dynamics of

bipedal motion, however, it does not account for the presence of feet on a real character model. When feet are flat on the ground, they afford the controller extra control potential through limited actuation of the root (previously assumed unactuated) ankle joint of the simple point foot model. The actuation is limited because large forces on the ankle joint would cause the feet to rotate on edge which would result in the character falling down.

To determine the allowable actuation of the root ankle joint, this section will define the concept of an FRI policy, which is an auxiliary policy that defines how the character uses the ankle joint advantageously, while avoiding foot to rotate. This section will only define the concept of the FRI policy and demonstrate how it is used in conjunction with the motion constraints. Design of useful FRI policies is a different matter all together. In the results section of this chapter, some simple FRI policies will be defined, but a large portion of Chapter 5 is devoted to the use of policy optimization to design better FRI policies.

To define the FRI policy, we must first define the notion of the foot rotation indicator point (or FRI). The FRI is defined as a point on the ground that must remain in the base of support of the foot to prevent rotation [29] (see Figure 3-3). Formally, FRI is defined as the point on the ground plane such that

$$\boldsymbol{u}_r = \text{FRI} \times \text{GRF}, \tag{3.11}$$

where $\boldsymbol{u}_r$ is the torque on the root ankle joint (and assuming a massless foot). For those familiar with the similarly purposed zero-moment point (ZMP), it may be convenient to realize that both the FRI and ZMP are identical when the foot does not rotate. Thus, for the purpose of the following exposition, they may be treated identically.

Under the feedback of the motion constraints, the instantaneous acceleration of the zero dynamics system is fully determined by the choice of $\boldsymbol{u}_r$ (cf. Equation 2.29). As the GRF is solely a function of the acceleration of the center of mass and we may derive the following linear relationship:

$$\text{GRF} = \boldsymbol{A}_{\text{GRF}}(\theta, \dot{\theta})\boldsymbol{u}_r + \boldsymbol{b}_{\text{GRF}}(\theta, \dot{\theta}), \tag{3.12}$$

where $\boldsymbol{A}_{\text{GRF}}$ and $\boldsymbol{b}_{\text{GRF}}$ are trivially computed by plugging in the reconstruction functions (3.6) and (3.7) into a equation that computes the ground reaction force. Of course the validity of this equation depends upon the action of the motion constraints ensuring that $\boldsymbol{y} = 0$.

In the 2D case, Equation 3.11 reduces to a simplified form:

$$\boldsymbol{u}_r = (\text{GRF}^y \text{FRI}^x - \text{GRF}^x \text{FRI}^y). \tag{3.13}$$

By substituting (3.12) into (3.13) and solving for $\boldsymbol{u}_r$, it can be seen that in order to prevent the foot from rotating, $\boldsymbol{u}_r$ must be limited to the set

$$\{\boldsymbol{u}_r = \frac{\text{FRI}^y \boldsymbol{b}^x_{\text{GRF}} - \boldsymbol{b}^y_{\text{GRF}} \text{FRI}^x}{\boldsymbol{A}^y_{\text{GRF}} \text{FRI}^x - 1.0 - \text{FRI}^y \boldsymbol{A}^x_{\text{GRF}}} | \text{FRI}^{x-} < \text{FRI}^x < \text{FRI}^{x+}\}, \tag{3.14}$$

where FRI^{x+} and FRI^{x-} are the upper and lower bounds of the support foot's contact with the ground. Rather than choosing a value of $\boldsymbol{u}_r$ directly, we define an FRI policy,

$$\Pi(\theta, \dot{\theta}) : \mathbb{S} \rightarrow [\text{FRI}^{x+}, \text{FRI}^{x-}] \in \mathbb{R}, \tag{3.15}$$

which maps from the reduced state $(\theta, \dot{\theta}) \in \mathbb{S}$ to a value of FRI in the valid range. Given a value of FRI from the policy, $\boldsymbol{u}_r$ is computed using equation (3.14) and then $\boldsymbol{u}_a$ is computed using equation (2.21).

This would complete the calculation of the desired joint torques if we choose to ignore frictional contact constraints that limit the allowable direction of the ground reaction force vector. This is the case for Westervelt and colleagues [64]. They make the reasonable assumption that if the disturbances to the character are small, ground reaction forces will remain valid due to the presence of gravity. Gravity ensures that ground reaction forces occur mostly in a vertical direction, thus avoiding pulling on the ground or large tangential forces. However, when disturbances to the character become larger, this assumption is easily violated. Since a real character cannot pull on the ground with it's feet, attempting to blindly execute the feedback from the motion constraints would cause the character to slip or lift its foot off the ground.

To prevent this, our controller has a mechanism for detecting when such situations will occur and handling them. Whenever the computed forces from the motion constraints and the FRI policy would result in a GRF outside allowable friction cones, our controller projects the GRF back onto the allowable friction cone. The projected version of the GRF is no longer consistent with the motion constraints, so a least squares minimization is solved to compute desired joint accelerations:

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}}} \quad & \|\ddot{\boldsymbol{y}}(\ddot{\boldsymbol{q}}) - \ddot{\boldsymbol{y}}^*\|_2 \\ \text{subject to} \quad & \text{GRF} = \text{GRF}^*, \\ & \text{FRI} = \Pi(\theta, \dot{\theta}) \end{aligned} \tag{3.16}$$

where GRF^* is the projected version of GRF and $\ddot{\boldsymbol{y}}^*$ is the desired acceleration of the motion constraints given by Equation 2.22. This optimization is formulated as a linear-equality constrained quadratic program and solved efficiently at runtime. Once $\ddot{\boldsymbol{q}}$ has been determined, an $O(n)$ inverse dynamics algorithm (Featherstone's Newton-Euler method [27]) is used to computes $\boldsymbol{u}$.

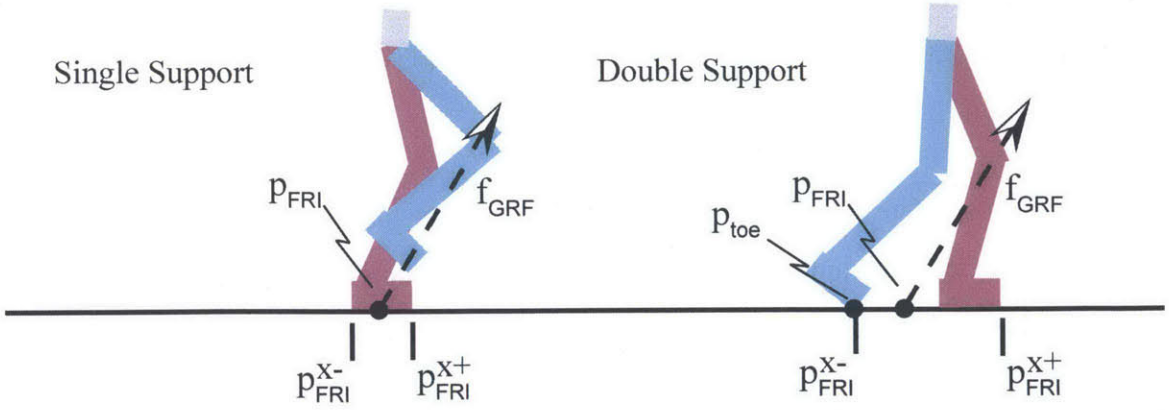

Figure 3-3: *The foot rotation indicator point FRI and ground reaction force GRF are depicted during double and single support. The horizontal component of the FRI, FRI^x, must remain within the bounds of the support $[FRI^{x+}, FRI^{x-}]$ in order to prevent foot rotation. Our use of the FRI is somewhat overloaded in the sense that we use it to describe valid placement of ground reaction forces during double support as well as single support. In the double support cases the FRI can be thought of as describing the placement of ground reaction force on a large virtual foot rooted at the ankle joint of the stance leg and extending to cover the base of support of both feet. This is valid because in double support the swing toe, $\mathbf{p}_{toe}$, is constrained to remain in contact with the ground.*

3.3.2 Double Support

The control of a biped during a non-instantaneous double support phase has not been addressed in prior work using motion constraints. Others [64, 51] have treated the double support phases as an instantaneous impulse event. This avoids the problem of dealing with closed loop kinematics during the double support stage of the motion, but it also limits motions to ones that don't often occur in nature. The walking gait of a human, for example, involves a non-instantaneous double support phase known as toe-off, where the back foot pushes off the ground, injecting forward momentum. This phase is critical to injecting energy into a gait when walking up steep inclines and also helps to make the gait more robust to force disturbances.

In double support, it is assumed that the toe of the back foot makes contact with the ground. To ensure that this is the case, even when disturbances occur, the motion constraints (3.3) are modified by selecting a new invertible coordinate transform,

$$\boldsymbol{g}^{-1}(\boldsymbol{q}) = [\boldsymbol{h}_u(\boldsymbol{q})^T, \boldsymbol{h}_a(\boldsymbol{q})^T]^T \tag{3.17}$$

$$\boldsymbol{h}_u = \theta, \qquad \boldsymbol{h}_a = [\tilde{\boldsymbol{q}}_a, \boldsymbol{p}_{\text{toe}}]^T, \tag{3.18}$$

where the tilde symbol represents an operation that removes elements from the vector correspond to the heel and knee of the second leg and $\boldsymbol{p}_{\text{toe}}$ is the Cartesian position of the toe of the second foot. The motion constraints are redefined to be

$$\boldsymbol{y} = \begin{bmatrix} \tilde{\boldsymbol{q}} \\ \boldsymbol{p}_{\text{toe}} \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{c}}(\theta) \\ \hat{\boldsymbol{p}}_{\text{toe}} \end{bmatrix}, \tag{3.19}$$

where $\hat{\boldsymbol{p}}_{\text{toe}}$ is constant value representing the desired position of the toe on the ground. This has the effect of adapting the motion constraints so that they explicitly enforce the ground contact constraint of the second foot. The same FRI control strategy is applied as in the single support case, except that the bounds on the FRI are expanded to include the inscribing polygon of both feet (see Figure 3-3).

In the final computation of joint torques, there is still an ambiguity due to redundancy in

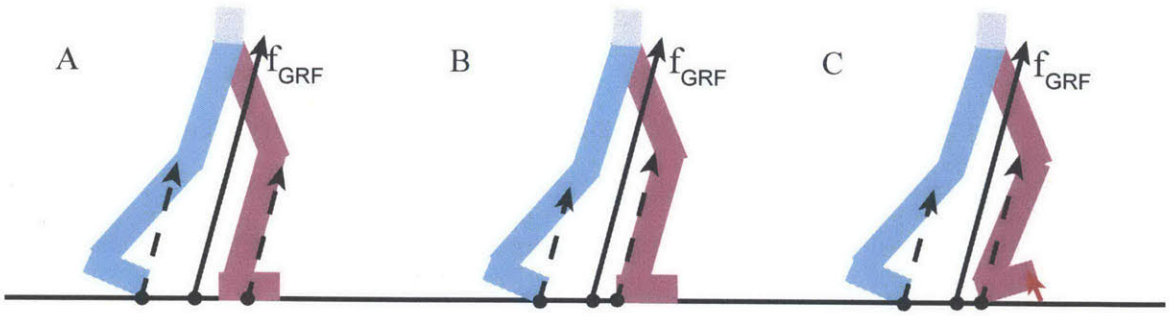

Figure 3-4: *This figure depicts the redundancy in how the aggregate ground reaction force may be applied to a simulated character in double support by varying the forces on each foot. In illustrations A and B, the same aggregate ground reaction force (solid arrow), originating from the same point on the ground, is the sum of different individual forces on each foot (dashed arrows). In figure A the aggregate force on each foot is placed strategically at the center of the foot. In figure B, the aggregate force on the front foot is placed on the edge of the heel. This is a less stable configuration because even a small disturbance (small red arrow, illustration C) would cause the front foot to rotate about the heel, reducing the effective area of support. On the other hand, a small disturbance to the front foot in A would simply move the point of origination of the foot force slightly toward the heel, while still keeping the foot stationary. Thus, placing the force at the center of each foot effectively results in a self-stabilizing contact configuration, which helps in cases when the foot briefly rolls on edge due to unmodelled disturbances.*

the actuation of the closed-loop configuration (see Figure 3-4). This ambiguity corresponds to a choice of where to place the aggregate linear force on each foot. Previous control strategies for characters have either ignored this redundancy or resolved it implicitly though a minimum joint torque criteria. However, we have found that a simple strategy that works well is to choose an aggregate force near the center of each foot. This corresponds to the intuitive idea that force should be placed away from the edges of the feet. Empirically we have found that this results in a control strategy that is more robust to unmodeled force disturbances that might otherwise cause the feet to rotate on edge .

We use an efficient algorithm for computing the final joint torques during double support. It involves 4 steps:

1. calcuate a desired set of accelerations $\ddot{\boldsymbol{q}}_d$ by solving the least squares minimization (3.16) subject to the motion constraints (3.19) and the FRI policy

2. using $\ddot{\boldsymbol{q}}_d$, calculate the joint torques, $\boldsymbol{u}_{\text{open}}$, for an open-loop skeleton rooted at the first foot using the Newton-Euler O(N) recursive inverse dynamics algorithm

3. determine the vectors $\boldsymbol{f}_0, \boldsymbol{f}_1$ and positions $\boldsymbol{p}_0, \boldsymbol{p}_1$ of aggregate forces on each foot, such that GRF $= \boldsymbol{f}_0 + \boldsymbol{f}_1$ and the location of the FRI remains the same (i.e., $\boldsymbol{f}_0 \times (\boldsymbol{p}_0 - \text{FRI}) + \boldsymbol{f}_1 \times (\boldsymbol{p}_1 - \text{FRI}) = 0$). if possible, place the $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ at the center of flat feet.

4. calculate the final joint torques as $\boldsymbol{u} = \boldsymbol{u}_{\text{open}} + (\frac{\partial \boldsymbol{p}_1}{\partial \boldsymbol{q}})^T \boldsymbol{f}_1$, which projects the aggregate linear force from the second foot, $\boldsymbol{f}_1$, back into the joint space through the Jacobian transpose

The control formulation is only slightly more involved than a standard inverse dynamics calculation and yields a similar $O(n)$ algorithm. The resulting torque is applied directly to the simulated character. Pseudo-code for a slightly simplified walking controller with only two stage (double support and single support) is provided in Algorithm 1.

Algorithm 1 The following pseudo-code is an example of the control algorithm which executes at each control interval for a simplified walking controller. The simplified walking controller only has two stages; one single support stage (SINGLE) and one double support stage (DOUBLE).

```
stage ← DOUBLE
loop
    θ, θ̇ ← ComputeTheta(q,q̇)
    if stage = DOUBLE && θ > θ_s then
        stage ← SINGLE
    else if stage = SINGLE && footContact() then
        stage ← DOUBLE
    end if
    FRI = Π(θ, θ̇)
    (u, GRF) = ComputeTorque(FRI)                    ▷ Eqn. (3.14) and (2.21)
    if GRF^y < 0 then
        GRF = 0
        u = ComputeLeastSquaresTorque(FRI, GRF)      ▷ Eqn. (3.16)
    else if ||GRF^x / GRF^y|| > fric. coef then
        GRF^x = sign(GRF^x) * GRF^y
        u = ComputeLeastSquaresTorque(FRI, GRF)      ▷ Eqn. (3.16)
    end if
    (q, q̇) = ForwardSimulate(q, q̇, u)
end loop
```

3.4 Results

Here we will discuss results of using the controller described in this chapter to create interactive simulations of a human biped. We start by describing the process of creating controllers for walking. It will be demonstrated that our walking controller is extremely robust to many different kinds of disturbances including uneven ground and force perturbations.

Walking is a particularly well studied motion type that will allow us to easily compare and contrast our controller method with others tracking [20, 45] and non-tracking [67, 48] gait controllers in graphics. One advantage of our controller over non-tracking controllers is the ease with which the style of the walk can be altered by exchanging one input motion for another. We also will show that our controller is significantly more robust that a state-of-the-art time-indexed tracking controller based upon the Nonlinear Quadratic Regulator. We will discuss advantages and disadvantages of each formulation.

Finally, we describe a jumping controller that we have constructed using the phase-indexed control approach. The jumping controller will demonstrate the potential of the phase-indexed control approach to handle motion types other than walking. The jumping controller will involve an underactuated take-off phase and an aerial flight phase that will necessitate slight modifications to the controller described so far.

3.4.1 Walking

To construct our walking controller we start with a recorded motion and apply a spline-smoothing technique [38] to produce a walk cycle. Next we identify a monotonic variable θ. In the case of walking, a particular convenient choice is the angle of the vector between the stance ankle and the hip. This choice resulted in robust gaits, however, we also experimented with using the horizontal position of the center of mass. This worked better for styles of walking where the angle of the swing leg was not monotonic or when the velocity of the swing leg varied too much over the course of a step. In general we found that the

larger the accelerations of the θ variable the less stable the resulting tracking controller. This is due to the larger forces that are generated in response to sudden changes in the variable when disturbances occur.

To create a walking controller we start with physically valid motion trajectory. Next, we partition the input motion into different support stages and fit the parameters of the motion constraints to the motion using the process described in Section 3.2.

The final step in controller design is to define an FRI policy Π. The simplest policy we tried keeps the FRI constant throughout the entire gait cycle. For some styles of walking, this policy was sufficient to achieve a constant walking speed. Moving the FRI forward (or back) resulted in slower (or faster) walks. If the FRI was brought too far forward, the controller came to a stop, or, in some case, took a backwards step, depending upon the parameters of the motion constraints.

A constant FRI policy will not robustly combat unexpected perturbation such as deviation from flat ground. A simple way to regulate the forward walking speed is to incorporate a PID controller. We found through experimentation that using the integral term alone works best:

$$\Pi(\theta, \dot{\theta}) = \min(fri^{+}, \max(fri^{-}, fri_0 + \int_t e(t)dt)), \tag{3.20}$$

where $e(t) = \dot{\theta} - \dot{\theta}_d$ is an error between the current and desired velocity of θ.

The walking controllers that we have built using this strategy can consistently withstand a forward or backward push to the upper torso of up to 350 Newtons for 0.1 seconds at all points along the walk cycle. These results are comparable to the ones described by other robust biped walking controllers (e.g., SIMBICON controller [67] which can withstand 600N for 0.1s) after accounting for the smaller weight of our character (51 vs. 90 kg).

Robustness

The walking controller we have designed are robust to variations in the terrain as well as unanticipated pushes applied to the body. The same controller designed to walk over flat ground also makes forward progress over a sloped ground between -18 and 10 degrees. The controller is able to sense when contacts are made, but has no notion of ground geometry. If the phase variable exceeds its range, the controller simply projects the variable back onto the closest value in the range in order to compute the virtual constraint. We experimented with other interesting terrain adaptations, including walking over stairs, a spongy ground, and a moving link bridge. Even without adjusting a single parameter, the basic controller for walking on flat ground proved remarkably robust to these ground perturbations. The main failure mode of the basic tracking controllers was when the toe unexpectedly stubbed the ground. One possible solution would be to check for ground clearance and switch to a controller with a higher step.

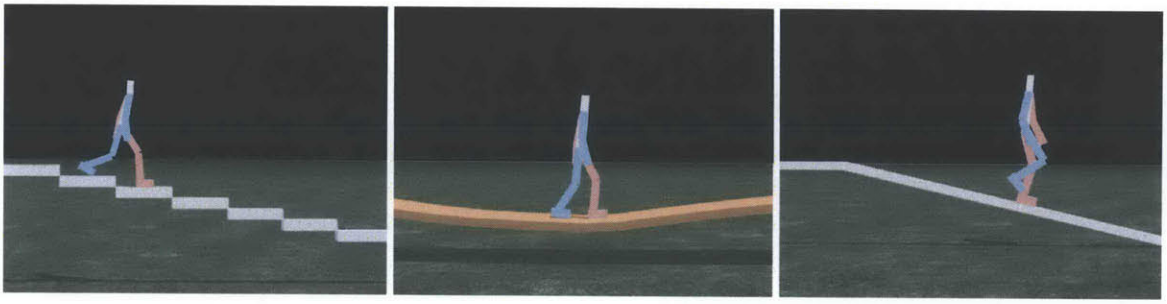

Figure 3-5: *Our phase-index walking controller designed for flat ground is remarkably robust, even when walking over unexpected ground surfaces, such as down steps or over a moving bridge.*

3.4.2 Comparison with Quadratic Regulator

The linear quadratic regulator (LQR) and nonlinear variants [20, 45] have shown promising results for tracking motions with robustness. They represent the current state-of-the-art in time-indexed trajectory tracking. For comparison purposes, we have implemented both our controller and a nonlinear quadratic regulator (NQR) on a simplified 5-link model with point feet.

To produce a trajectory for the NQR to track, our controller is run for six steps. Design of the NQR involves tuning 5 different parameters per model degree of freedom (DOF). To simplify the tuning process the same 5 parameters are used for all DOFs. Both controllers are tuned to be as robust as possible while retaining some compliancy. Torques are clamped to 300 Newton meters to prevent use of large forces. To test the robustness, a force is applied to the torso midway through the second step.

After tuning the NQR gains for maximum stability the character is able to withstand forces from -50 to 10 Newtons (in the horizontal direction for 100 milliseconds) without falling down during the remaining steps. By contrast, our controller could sustain forces between -500 and 400 Newtons. Table 3.1 summarizes parameter sensitivities of the two controllers showing the viable range of parameters than could recover from the push.

Qualitatively, the response of our controller to the -50 Newton meter push is notably different to that of the NQR controller. The NQR controller flails a leg outwards in order to catch up with the original timing of the motion. The motion will often diverge significantly from the original trajectory which can create exciting, dynamic recoveries. However these recoveries are not always natural or graceful. We found that the exact response depends heavily upon the setting of the parameters in a unintuitive manner. For example, increasing the penalty for deviating from the reference value of the joint angles had the opposite effect in some case. Our controller recovers in a more predictable manner, while using smaller torques to do so (see Figure 3-6).

Table 3.1: *Our Controller: Parameter range for stable response to a -50 Newton push.*

param	value	min	max	description
ε	0.05	0.001	0.49	exponential convergence factor
K_s	10	1e-2	1e4	position and velocity gain ($K_d = 2\sqrt{K_s}$)

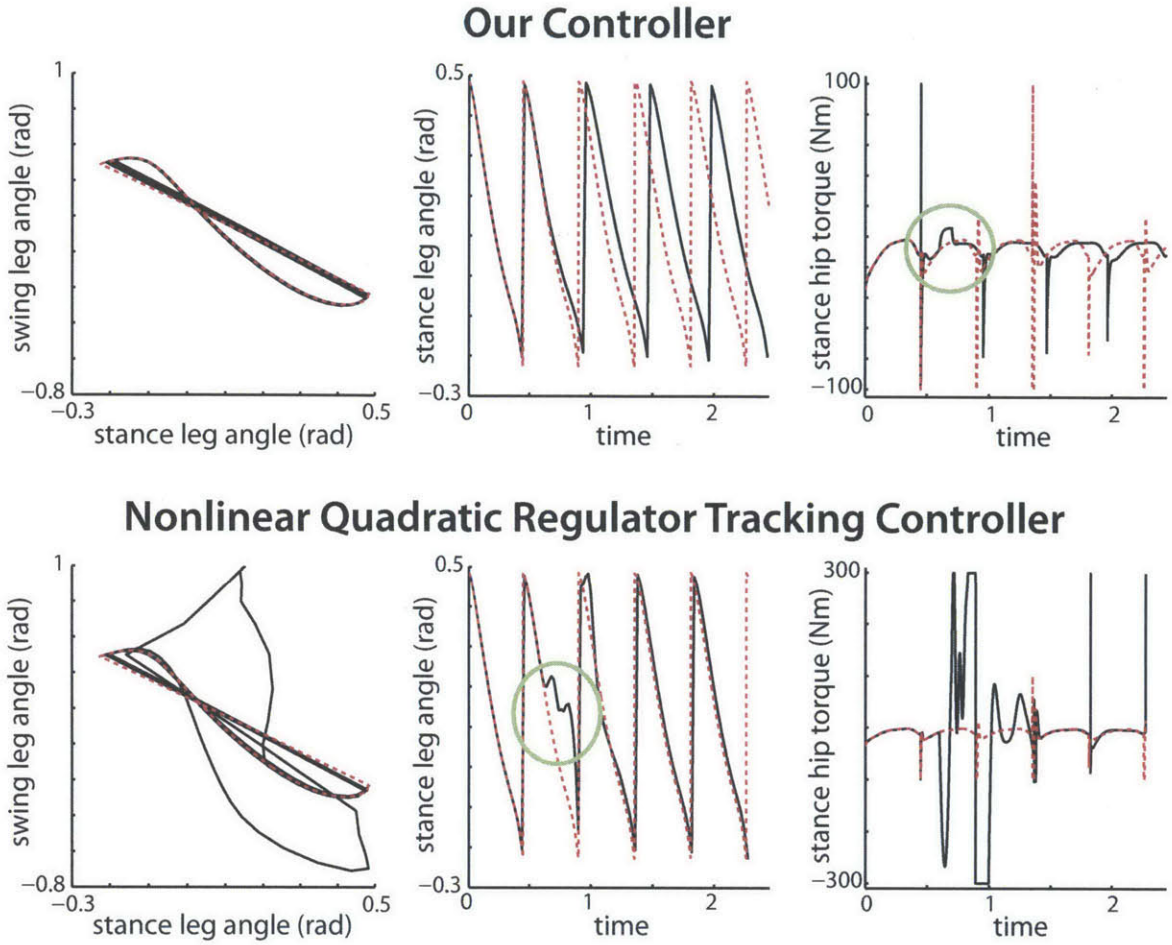

Figure 3-6: *Graphs comparing the response of our controller vs. the NQR controller (to a push of -50 Newtons for 100ms). Black lines indicate the response and red lines are the unperturbed reference trajectory. Our controller stays close to the original trajectory (left) by deviating from the original timing (middle). The NQR controller closely tracks the original timing (middle), but uses larger torques (right) to recover. Compared to the NQR controller, our controller can recover from pushes that are an order of magnitude larger.*

3.4.3 Broad Jumping with Dynamic Balance

Our broad jump controller performs a sequence of forward jumps. This motion involves dynamics balance, where the center of mass is not over the base of support, making it tricky to perform. The key control challenge is in regulating the speed of the landing so the character is prepared for the next jump. Unlike the case of walking, a simple FRI policy fails to result in a self-stabilizing motion. Instead an FRI policy is designed using a discretized value iteration on the reduced state.

Table 3.2: *Nonlinear Quadratic Regulator: Parameter range for stable response to a -50 Newton push.*

param	value	min	max	description
Q	1e6	0	1e7	position cost
$\dot{Q}$	1e2	6e1	6e4	velocity cost
Q_{end}	1e8	1e7	1e9	final position cost
$\dot{Q}_{end}$	1e8	1e2	1e8	final velocity cost
R	1e1	1e0	1e4	actuation cost

For this motion, the monotonic variable θ is the horizontal position of the center of mass relative to the feet. The jumping controller is divided into 4 stages (INIT, TOEOFF, FLIGHT, LAND) corresponding to different contact configurations between the character's feet and the ground (see Figure 3-7). Switching between stages occurs at designated values of θ or, in the case of the transition to LANDING, when the feet are flat on the ground.

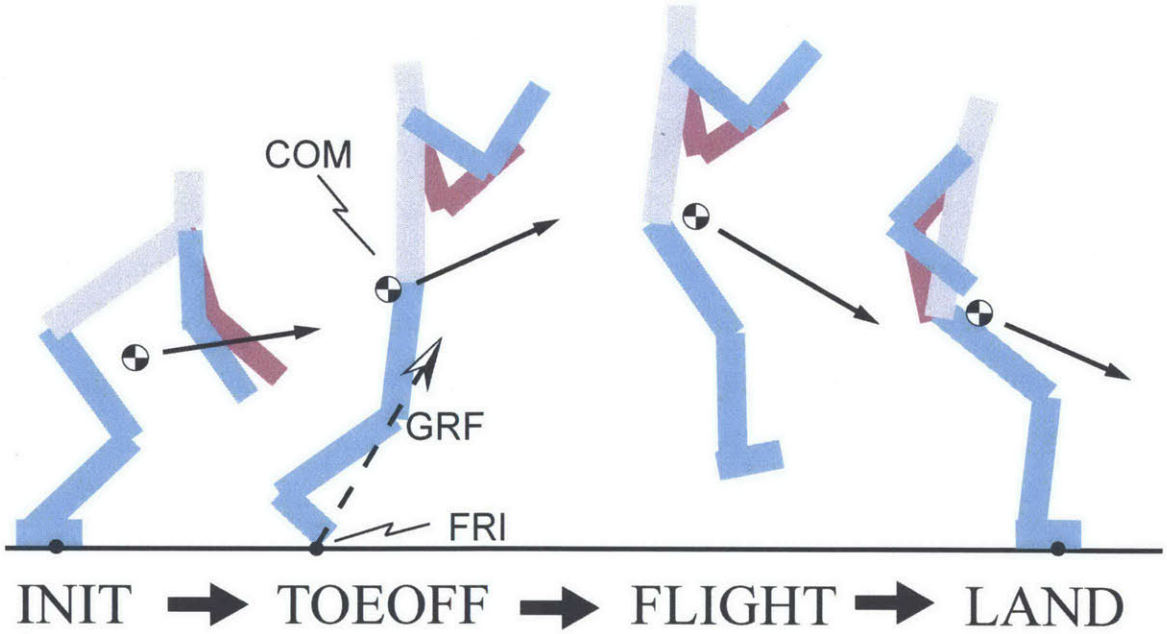

Figure 3-7: *Take-off initiates with the character feet flat on the ground. Briefly before flight, the character pushes off with its toes. In the flight phase there is no contact. The character enters the landing phase when the feet are flat on the ground again. Finally, the character transitions back to takeoff.*

During the FLIGHT phase, the initial angular and linear momentum fully determines the trajectory of the character. As such, the controller is highly sensitive to the ground reaction force through the TOEOFF stage. To better control these forces, motion constraints on the left knee and ankle are replaced with direct control over the ground reaction forces. In the design phase, we fit a spline function $\text{GRF}(\theta)$ to the feedforward reference values of these forces, as a function of θ. Motion constraints in the FLAT, FLIGHT, and LAND phases are treated the same way as in the double supports stage of walking.

The jump controller is particularly sensitive to the location of the FRI throughout the LAND and INIT stages. Depending upon the value of the FRI, the jump will either speed up or slow down resulting in the character falling down after several cycles. Simple FRI policies, such as the integral controller described for walking, do not work. Instead, we design an FRI policy using value iteration (Section 5.2.1).

3.5 Comparison to Related Work

The work of Westervelt and colleagues [64], in particular, was influential in the development of our phase-indexed controller. Our work is inspired by theirs and many of the specific design choices are the same. These choices include the use of the stance leg orientation as a monotonic phase variable for walking, the specific form of the constraint stabilizing feedback (Equation 2.22), and the use of inelastic collision dynamics (Equation 3.9) to model the effect of foot to ground contact events. Westervelt and colleagues have taken the analysis of these concepts to great depths and have shown precisely the conditions under which virtual constraints produce provably stable passive gaits. Moreover they have empirically verified these analyses by running their controller on a planar biped robot with point feet.

Although the primary focus of their work has been on the study of passive gaits for bipeds with point feet, they have also described two extensions for actuating the ankle of bipeds with feet. One extensions they describe controls the evolution of the FRI point path in a feedforward manner, effectively shaping the passive zero dynamics, without perform-

ing any active feedback [12]. The other extensions is similar to the simple proportional-integral controller described in this chapter [64],. However, a key distinction between their approach and ours is that they use feedback on the ankle actuation only to improve the convergence rate of gaits which are already passively stable. By contrast, many of the gaits we have experimented with are not passively stable. Instead, our controllers rely upon the actuation afforded through an FRI policy to encourage convergence to a limit cycle. We find this approach works well for most anthropomorphic gaits we have experimented with.

A major limitation of the controllers described by Westervelt and colleagues is the lack of a finite-duration double support phase. In their controllers, switching between the swing and stance legs occurs at an instantaneous impulse event. This is limiting since most anthropomorphic gaits exhibit some finite period of time when both feet are on the ground. In fact, it is likely that the extra actuation potential afforded by double support is partially responsible for the success of our FRI policy in producing stable gaits, despite the lack of passive stability. Thus, a clear implication of our work is that anthropomorphic gaits with double support need not be passively stable. In fact, it is likely that many anthropomorphic gaits, of interest for animation purposes, are not passively stable.

We have compared the stability of our phase-indexed controller to a time-indexed controller based upon the NQR formulation[45]. Base upon our results (see Section 3.4.2) we have argued that our controller performs better than the time-indexed controller because it doesn't try to maintain a specific motion timing. Although our controller uses the positioning of the FRI to correct the velocity state, it lacks the ability to perform full body corrective motions. Studies of actual human balance have typically identified two types of balance strategies: an *ankle strategy* and a *hip strategy* [30]. Our control is capable of the first, but not the second. Using our controller, a bipeds without an ankle or a double support phase cannot perform any continuous velocity correction of the underactuated degree of freedom over a single step. Instead, corrections to the velocity for point feet bipeds must be performed over the course of several steps by taking appropriate transitions. The NQR controller was able to perform these sorts of corrections, but did so at the expense of being significantly less stable in response to force disturbances.

By contrast, a different kind of phase-indexed controller, based upon a linearization about transverse coordinates [51, 42], is able deviate from the motion constraints in order to perform whole-body corrective motion, while not requiring strict timing. Like the NQR controller, transverse coordinate-based controllers rely upon a linearized model of the character's dynamics, but these controllers do so without a motion clock, similar to our approach. These controllers serve as an interesting middle ground between our approach and the NQR approach. An interesting avenue of future work would be to compare this approach to ours in terms of robustness and motion style. It is likely that both approaches are appropriate in different situations. For example, for long or slow continuous motions it is probably better to attempt to correct velocity state by temporarily deviating from the motion constraint surface (e.g., hip strategy). However, for motions with frequent, discrete motion transitions it may be unnecessary to perform such corrections in a short duration, making the phase-indexed controller a simple and effective alternative.

Chapter 4

Motion Rectification and Editing

One key goal of the work presented in this thesis is to suggest a tool by which animators can easy create stylistic controllers for simulated characters. However, the phase-indexed controller described in Chapter 3 relies upon the assumption that the input motion being provided is physically feasible in a strict mathematical sense. This is because a motion which is not physically feasible will be impossible to perform in simulation. Unfortunately, it is usually impossible to manually detect whether this is the case for a given motion and even harder to manually correct if it is not. This is somewhat at odds with the goal of providing an easy process for animators. Thus in this chapter we describe an optimization approach which semi-automatically transforms an a motion which is physically plausible into one that is physically feasible.

The approach taken is to allow animators to create stylistically desirable motion through whatever means they choose. Then, an optimization is solved that minimally modifies the motion, with user guidance, to ensure physical feasibility. We call this process motion rectification. In other words, motion rectification is the process of starting from an input motion and automatically finding a similar motion which is physically feasible.

The optimization we solve is in the form of a trajectory optimization (see Section 2.2.3) . In addition to rectifying the original motion we are also able to slightly modify the optimization to create systematic edits to a motion. We call this editing process motion editing.

Using these edited motions, it is possible to create entire parameterized switching controllers (Chapter 5), starting from only a single input motion.

Speed is a key consideration in the design of our trajectory optimization algorithm because of the importance of allowing iterative refinements to the solutions through user guidance. Due to this consideration, we have employed a gradient-based optimization approach. Gradients allow a search direction to be calculated quickly, without the need for time consuming point sampling. The down side of this approach, however, is that gradients can be difficult to calculate robustly. Thus one of the key contributions of the following algorithm is a robust approach to computing the gradient of important dynamics constraints which ensure that external forces acting on the character remain within physical bounds.

This chapter will start by discussing the components of the core algorithm used to perform both rectification and editing. Then variants of the core algorithm for each process will be discussed, followed by examples of how the algorithm is used to generate reference trajectories for our phase-indexed controller. Finally a discussion of key considerations in the design of our algorithm will be presented.

4.1 Trajectory Optimization using Direct Transcription

We solve a trajectory optimization problem using a direct transcription method (see Section 2.2.3). Using this approach, a continuous motion is represented using a discrete time series, $\boldsymbol{x}_i = (\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$, of the character's state. An optimization problem is solved that minimizes a performance metric $p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots \boldsymbol{x}_n)$, subject to problem constraints $\boldsymbol{c}_j(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots \boldsymbol{x}_n)$. The problem can be formulated as a sparse, nonlinear program (NLP). The NLP is solved using the sequential quadratic programming package SNOPT [28].

4.1.1 Constraints

Constraints on the character motion, $\boldsymbol{c}_j$, fall into three categories: kinematic constraints, collocations constraints, and physicality constraints. The kinematic constraints ensure that

the placement of the feet are well aligned with the ground and that the monotonic θ variable, used by the phase-indexed controller, remains strictly increasing. Collocation constraints ensure that the time series solution is consistent with a desired numerical differencing scheme. This ensures the correct relationship between the joint positions, $\boldsymbol{q}_i$, and velocities, $\dot{\boldsymbol{q}}_i$. Physicality constraints ensure that the forces acting on the character are physically consistent. As opposed to standard direct transcription formulations, which include the external and internal forces explicitly as free variables in the optimization, our approach completely ignores internal forces and only calculates the external forces on the character as part of a physical constraint. The physical constraints ensure that the external forces originate from within regions of contact between the feet and the ground and that these forces lie within the coulomb friction cone of the contact surface to ensure that the frictional contact dynamics are not violated. The external forces are calculated by finite differencing generalized momentum, which can be calculated directly from the state at each sample point along the trajectory.

Both the kinematic and physicality constraints require some initial information about the timing and placement of the feet on the ground. In addition to the original motion data, our problem formulation takes as input a time-based segmentation of the motion data that describes how feet are in contact with the ground. There are 4 possible contact modes for each foot: HEEL, TOE, FLAT, and FREE. HEEL and TOE correspond to point contacts at a desired location on the ground, FLAT includes information about both orientation and position of the contact, and FREE means the foot is not in contact. The timing of this segmentation is provided manually by the motion designer, but the location of the contacts are calculated automatically by projecting the location of the feet from the input motion onto the ground surface, at the beginning of each segment. In this step, we also compute a 1D convex hull, $(\boldsymbol{h}_i^+, \boldsymbol{h}_i^-)$ of the region of contact between the feet and the ground. This is used to express the constraints on ground reaction forces more compactly.

In totality, the constraints are:

Kinematic Constraints

$$c_j = p_j - \hat{p}_j = 0, \tag{4.1a}$$

$$c_j = (\theta_{i+1} - \theta_i)/\Delta t \geq k_\theta, \tag{4.1b}$$

where the subscript j denotes the index of the constraint, p_j is the current Cartesian location of the contact point on the foot and $\hat{p}_j$ is the desired location on the ground.

Collocation Constraints

$$c_j = (\boldsymbol{q}_{i+1} - \boldsymbol{q}_i)/\Delta t - \dot{\boldsymbol{q}}_i = 0, \tag{4.2a}$$

where the subscript j denotes the index of the constraint and Δt is the finite difference between trajectory samples.

Physicality Constraints (when in free-flight)

$$c_j = \tau(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \quad =0, \tag{4.3a}$$

$$c_j = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \quad =0, \tag{4.3b}$$

Physicality Constraints (when in contact)

$$c_j = \tau(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) - (\boldsymbol{h}_i^- - \ell_r) \times \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \quad \geq 0, \tag{4.4a}$$

$$c_j = (\boldsymbol{h}_i^+ - \ell_r) \times \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) - \tau(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \quad \geq 0, \tag{4.4b}$$

$$\theta \leq c_j = \tan2(f_y, f_x) - \Delta\alpha \quad \geq -\theta, \tag{4.4c}$$

$$c_j = f_y - k \quad \geq 0, \tag{4.4d}$$

where the subscript j is the index of the constraint, τ is the root torque acting on the character, $\boldsymbol{f}$ is the root force acting on the character (subscripts y and x denote vertical and horizontal components), ℓ_r is the Cartesian position of the character's root, $\Delta\alpha$ is the angle of the ground plane, θ is the half angle of the coulomb friction cone, and k is a small positive constant. The purpose of the last constraint is two fold. It enforces a strictly

positive ground reaction force and also prevents the second to last constraint from becoming singular when both f_x and f_y become small.

4.1.2 Performance Metric

In the trajectory optimization problem, the goal of the performance metric is to encourage solutions that are close to the input motion. The performance metric is compost of several different terms that form a weighted sum:

$$p(\boldsymbol{x}_0,\boldsymbol{x}_1,\ldots\boldsymbol{x}_n) = \sum_{j=1}^{n} w_j p_j \tag{4.5}$$

As with the problem constraints, both value and the derivatives must be computed for all terms p_j. The following table summaries all terms and the range of respective weights, w_j, used.

Joint Angles	$(w_j = [0.1, 1.0])$	$p_j = \sum_i (\hat{\boldsymbol{q}}_i - \boldsymbol{q}_i)^2$ is the difference between the current joint angles and the those from the input motion, excluding root degrees of freedom.
Joint Velocities	$(w_j = [0.1, 1.0])$	$p_j = \sum_i (\hat{\dot{\boldsymbol{q}}}_i - \dot{\boldsymbol{q}}_i)^2$ is the difference between the current joint velocities and the those from the input motion, excluding the root degrees of freedom.
Joint Acceleration	$(w_j = 0.1)$	$p_j = \sum_i (\dot{\boldsymbol{q}}_{i+1} - \dot{\boldsymbol{q}}_i)^2/(\Delta t)^2$ is the discrepancy between subsequent joint velocities. This term encourages small accelerations which keeps the motion smooth and prevents large forces from being used. A term is omitted from the sum above whenever impulsive contact occurs between subsequent motion samples. Impulsive contact is assumed to occur between different segments of the motion where a new contact begins.
Torso Orientation	$(w_j = [0, 100.0])$	$p_j = \sum_i (\hat{\theta}_i - \theta_i)^2$ is the difference between the current global orientation of the torso and that from the input motion. This term prevents large deviations of the torso orientation, which are usually bad stylistically.
Swing Foot Height	$(w_j = [0, 100])$	$p_j = \sum_i (\hat{h}_i - h_i)^2$ is the difference between the current height of the swing foot and the height of the swing foot from the input motion. The input motion height $\hat{h}_i$ is offset by the displacement of the ground surface if the motion is being modified for a different ground surface.

4.1.3 Cyclic and Symmetric Trajectories

Most gaits, such as walking and running, are cyclic. Additionally, most gaits are symmetric about the left and right sides of the body. Therefore, when solving for a cyclic and symmetric motion trajectories, only the motion for the left or right stance needs to be included in the optimization. This reduces the number of variables by a half. Additional constraints and performance metric terms must be added to the problem to ensure that the motion is cyclic. For example, a collocation constraint,

$$c_j = (s(\boldsymbol{q}_0) - \boldsymbol{q}_{n-1})/\Delta t - \dot{\boldsymbol{q}}_{n-1} = 0, \tag{4.6}$$

ensures that the velocity of the final motion sample is consistent with the first sample, where s is an operator that swaps the left and right joint angles symmetrically and n is the number of motion samples. Similarly, an additional performance metric term ensures that the accelerations are smooth between the first and last sample:

$$p_j = \sum_i (s(\dot{\boldsymbol{q}}_0) - \dot{\boldsymbol{q}}_{n-1})^2/(\Delta t)^2 \tag{4.7}$$

When optimizing cyclic and symmetric motions, it is assumed that the input motion is also cyclic and symmetric. Input motions that are close to this, but may have slight discrepancies, such as motions coming from motion capture of an actual human, are processed to ensure that they are exactly cyclic. This is performed as a preprocess that smoothly blends the end of the motion into the beginning over the course of the motion.

4.1.4 Motion Transitions

Generating transitions between different motions is also a useful process. Instead of using an input motion, two existing cyclic motions that have been previously optimized are used

as input. The first motion is blended into the second using a using a smooth weighting function. It is assumed that the segmentation of the two motions is the same. However, if this is not the case, a segmentation can be provided to the system. Similar to the cyclic and symmetric case, additional constraints and objective terms must be added (or replaced) in the basic problem formulation in order to ensure that the end and beginning of the motion transition smoothly into and out of the two input motions.

4.2 Results

In this section we will discuss some the motion results obtained for the both trajectory rectification and editing.

4.2.1 Rectification

We were able to rectify 4 types of motion: a jumping motion, a running motion, a walking motion and a standing motion. For walking, we performed rectification on 3 different gait styles. In all cases, we used a 2D character model with 14 degrees of freedom. The input to the system was a 2D motion generated by pre-processing motion capture data of a human subject [52]. The duration of each motion varied between 0.67 and 1.5 seconds, however, the final sampling rate of each motion varied between 30 Hz and 90 Hz, depending upon how difficult it was for the solver to converge on a solution.

In most cases the optimization ran to convergence (or failure) within less than half a minute time and often much faster, on a mid-range desktop computer. This quick turn around time proved crucial for tuning the parameters of the optimization. The main parameters that required tuning were the sampling rate and the weights of various performance metric terms. From an artistic perspective, the main parameters that determined the final outcome where weightings on the torso orientation and swing foot height objectives. Turning either of the terms up too high could cause numerical convergence problems, however, turning them too low resulted in problems as well. Too little weight on the torso orientation often

caused the rectified motion to exhibit hyperactive torso motions that looked unnatural. Too little weight on the foot height objective often resulted in the swing foot interpenetrating the ground. Adjusting these motions was necessary to obtain a natural looking motion.

4.2.2 Editing

We were able to edit walking motions in three different ways. We changed the length of the step, the height of the step, and the duration of the step. Additionally, we were able to adjust the angle of the ground plan to create steps up and down a ramped surface. Changing the length and height of the step was accomplished by adjusting the kinematic motion constraints on the motion appropriately. Changing the duration was accomplished by scaling the time-step between samples. These adjustment are performed programmatically by a preprocess in our solver. The user of the system, provided three additional scalar parameters specifying the desired step height, percentage of the original step length, and percentage of the original step duration. The system is also able to automatically generate a family of edited motions by regularly sampling all three parameters simultaneously, within desired ranges. Once a family of different cyclic steps has been generated, the system can automatically synthesis transitions between these steps, creating a connected graph of physically valid motions. See Figure 4-1 for examples of the output from the editing process.

4.3 Discussion

The trajectory optimization presented here tries to balance accuracy and efficiency in order quickly generate trajectories that can be reliably tracked by the phase-indexed controller of Chapter 3. We used a simple finite difference scheme for all optimizations. One advantage of the finite difference scheme was that it allowed for simple handling of impulsive contact events, without resorting to modeling these events separately. We simply removed the contributions to the objective on smoothness over steps where impulsive contact was expected

to occur. This allowed for sudden changes to the velocity to occur over a single step in order to satisfy a changing set of contacts between the character and the ground. We were also able to use a relatively large sample frequency (e.g, 30-90 HZ). In general, we found that using a time-step that was too small resulted in numerical problems due to the division by the small Δt.

We tried several different formulations of the trajectory optimization problem before settling on the one presented here. In initial experiments, we tried including the physical constraints in the objective term, effectively transforming the problem into a unconstrained minimization. However, we found that we had to increase the objective weight on various constraint terms to the point where the problem became poorly scaled. By contrasts, the SNOPT solver handles such scaling problems in a principled manner. In fact, the SNOPT solver will remove constraints and add them to the objective dynamically when the problem becomes infeasible or poorly conditioned [28]. This works much better than our heuristic approach and truly speaks to the necessity of having an optimization code that handles variable scaling and constraints in a principled manner.

Since the end goal of optimizing the motions is to create motion constraint splines for the phase-indexed controller, we also tried optimizing the parameters of the constraints (control points of B-Splines) directly. Success performing trajectory optimization directly on a spline representation has been reported in the literature [64, 40] and our first impulse was to try this, rather than the less direct method of fitting a spline to the sampled solution after the fact. However, we found that we had more trouble with the NLP becoming infeasible when we optimized spline control points directly. When using splines it is more difficult to understand the relationship between control points and collocation constraints. Therefore it is more difficult to determine which constraints are causing a given optimization to fail. For most splines, the contribution of a given control point to the various derivatives of the spline will change depending upon the index of the derivative, the time index, and on the number of control points used, which only serves to further complicate constraint debugging. The use of a spline does not even reduce the number of free variables in the optimization, since in either case it must be ensured that there are enough degrees of freedom to satisfy all

active constraints. At least for our purposes, we determined that using finite samples as the free variables in the simulation was superior to optimizing spline control points directly.

4.4 Contributions and Comparison to Prior Work

Our trajectory optimization algorithms is most similar to the optimal motion synthesis algorithm of Fang and Pollard [26]. In accordance with their algorithm we have reformulated the direct transcription method to only include constraints on aggregate, full body forces and moments. As described in their work, it is possible to compute only these terms, independent of the internal joint actuations, reducing the number of free variables in the optimization, and the time it takes to compute the values and gradients. Fang and Pollard show this results in reduced optimization times, which is consistent with our experience as well.

Unlike Fang and Pollard, we further simplify the computation by only approximating aggregate forces by finite differencing aggregate momentum. Our experience indicates that this approach results in a more robust NLP that fails to converge less frequently. We suspect this is due to the fact that the momentum calculations do not involve the square of velocity terms (due to Coriolis and centrifugal forces) that could lead to greater numerical sensitivity.

In the work of Fang and Pollard, the aggregate linear force on the body (a.k.a., center of pressure) is constrained to a fixed point and no friction cone is enforced. This simplifies the constraints in the optimization and is common practice when optimizing motions for direct playback (see [49, 56]), presumably because the friction cone violations are avoided by other means or are small enough to be inpreceptible. By contrast, our motion must be feasible within simulation, thus necessitating precise friction constraints.

One way to enforce friction constraints in a trajectory optimization is to include additional force basis vectors as free variables in the optimization (i.e. [39]). This properly constraints the direction of the aggregate force, but increases the number of variables in the NLP. In

our optimization, we allow the location of the aggregate linear force (FRI) to vary over a support region, as well as enforce frictional contact constraints, but do so without increasing the size of the NLP. However, these constraints must be formulated carefully to prevent poor conditioning due to division by small numbers. This can occur, for example, when the aggregate linear force is significantly smaller than the aggregate angular moment. When this occurs, small changes in the angular momentum result in large jumps in the center of pressure, which can lead to numerical instability. Further more, when the aggregate linear force is zero, the center of pressure is not well defined. We have presented a set of constraints that avoid these situations by formulating the center of pressure constraints in terms of bounds on the aggregate forces rather than directly computing the location of the center of pressure (Equations 4.4b and 4.4c), and by preventing the magnitude of aggregate linear force from becoming too small (Equation 4.4d).

4.5 Conclusion

This chapter has described a method for rectifying and editing motion using a trajectory optimization with a direct transcription approach. The resulting optimization can be efficiently solved, at nearly interactive rates, using a well implemented sequential quadratic programming code. Using the methods described here, we were able to generate entire motion families from a single input motion, which we used as reference motions for our phase-indexed controller.

One deficiency of our trajectory optimization approach is that it does not allow for changes in the timing of the different segments of the motion. It is conceivable that certain motion would not be rectifiable without changing the timing, or that more extreme edits to motions could be performed with timing as a free variable. However, it is difficult to allow changes in timing using gradient-based trajectory optimization, since integrations schemes are particularly sensitive to the Δt parameter. A stochastic search scheme might be employed [62], but this would greatly increase the time it takes to optimize motions. Further more, it is likely the objective function would have to be redefined to allow differences in the timing

of the trajectory.

We did not try to use our rectification scheme on motions that clearly violated physics. There would be little point to trying to rectify such motions, as the motions would be impossible to simulate anyway. In general, the input to our solver must be close to physical. However, our solver was robust enough to handle motion capture data from a human with different dimension than the actual character model, even after that data was edited using simple linear blending techniques that don't preserve physical properties of the motion.

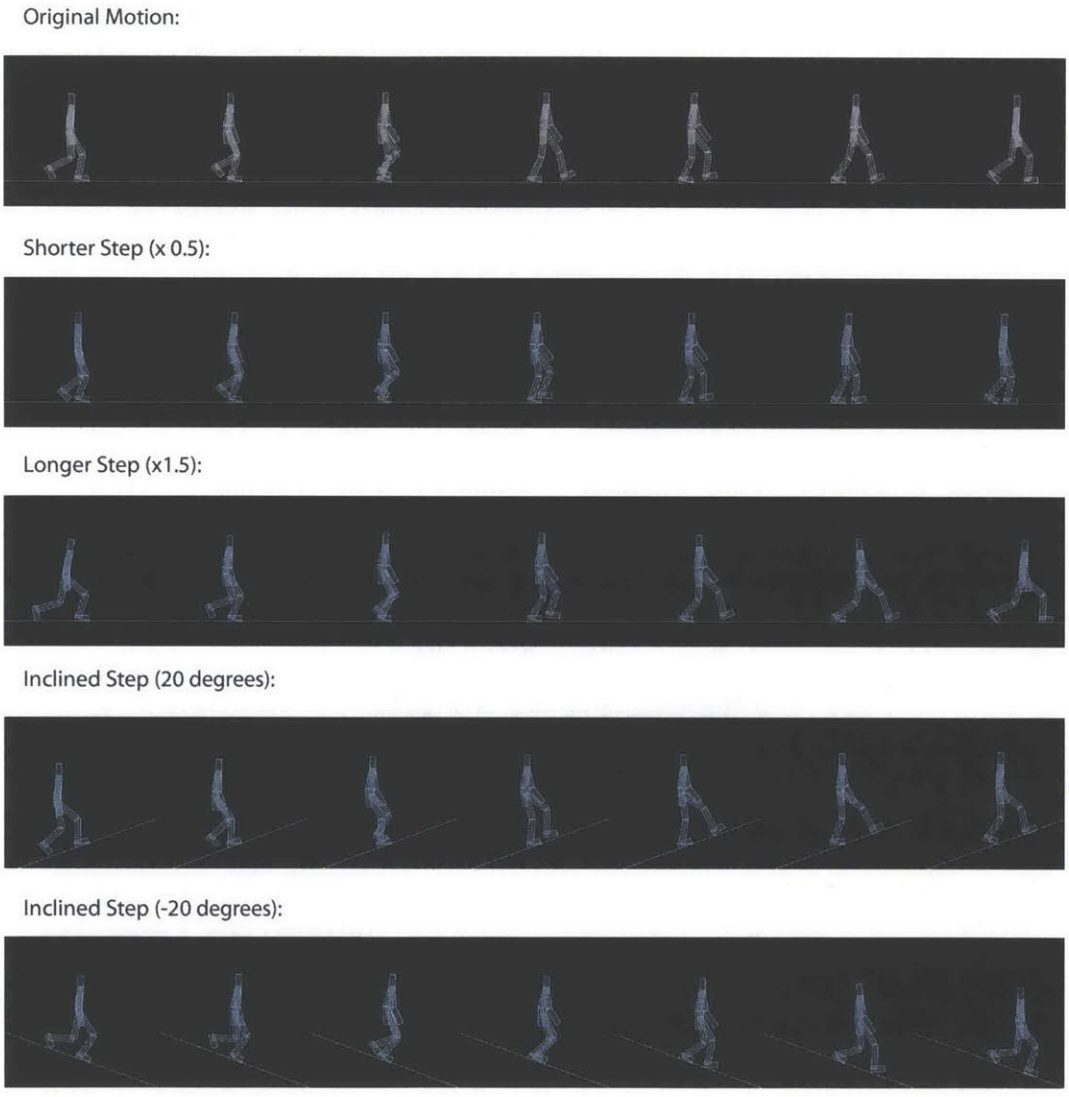

Figure 4-1: *This figure depicts a basic walking motion rectified and edited in a number of different ways.*

Chapter 5

Optimizing Policies using Value Iteration

Robust controllers must incorporate prediction so that the best action can be taken to accomplish future goals. Prediction must occur at many time-scales. Consider a walking controller. At the instantaneous time-scale, it must ensure that the foot does not slip. At the time-scale of an individual step, the controller must guide the joint angles along a prescribed path while rejecting unanticipated disturbances. At the time-scale of multiple steps, the controller must ensure that the character reaches its final destination while avoiding obstacles. The high-dimensionality of a character makes reliable prediction difficult to achieve in practice.

A key advantage of motion constraints is that they restrict the state of the character to the zero dynamics set (Section 2.2.2). Due to the low-dimensionality of this set it is computationally feasible to directly sample and tabulate the outcome of actions to provide predictions of future states. These predictions are in the form of transition functions,

$$\mathbf{T} : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S}, \tag{5.1}$$

that map a reduced state, $s \in \mathbb{S}$ and an action, $a \in \mathbb{A}$, to a future reduced state. Based upon these transition functions it becomes possible to optimize policies that accomplish goals in an approximately optimal fashion. In particular, we demonstrate the use of value iteration (see Section 2.2.3) to design optimized FRI policies for the phase-indexed controller as well

as policies for switching between different sets of motion constraints. The key insight of this chapter is to demonstrate how policies learned using the low-dimensional transitions functions can be beneficial for controlling the full character model, under the action of motion constraints.

5.1 Related Work

Value iteration is a well established method in robotics and other fields for designing optimized control policies. Many have applied it to the control of low-dimensional biped models [9, 14, 6, 44, 22] because computations become intractable in higher dimensions. One contribution of this thesis is to demonstrate how low-dimensional value iteration may be applied to more complicated bipeds using motion constraints. Our work is closely related to the use of reinforcement learning to improve the gait of a passive bipedal walker [59]. This work is not the first to apply reinforcement learning techniques to the problem of motion control for animated humans. It has been previously used to develop both kinematic [60] and dynamic [17] controllers that exhibit intelligent prediction. In both the case, a reduced representation of the state-space was used to make computation of the policy feasible. In the dynamic case, a carefully chosen reduced state was chosen but the dynamics of the reduced space is not known. In order to generate a transition functions, a complex example-based regression technique was required. In contrast, the approach described here computes the policy on the zero dynamics set where the dynamics can be precisely computed, without resorting to regression.

5.2 Policies

We demonstrate two types of optimized policies for phase-indexed controllers using value iteration. The first type of policy we demonstrate is a continuous FRI policy, for determining the continuous placement of the FRI point during a motion. This policy is evaluated at every control interval and is designed to regulate the speed of the motion such that the

controller achieves a steady state or an ideal transition velocity.

The second type of policy we demonstrate is a discrete step sequencing policy. Given a connected graph of possible stepping controllers, this type of policy determines which step controller to transition to in order to accomplish certain goals. The individual controllers that this type of policy acts on are the phase-indexed controllers designed using the motion rectification and editing method of Chapter 4. These individual controllers may rely on an FRI policy optimized independently of the discrete stepping policy. In the following sections, the method for optimizing both types of policies will be described in more detail.

5.2.1 FRI Policies

In this section we describe the method used to designing FRI polices for walking and jumping.

FRI Policies for Walking

The goal of the FRI policy for walking is to shape the speed profile of a stepping motion and to ensure that it does not diverge toward a failure state. For some motions, a simple FRI policy, as described in Chapter 3 (Equation 3.20), is sufficient for controlling the speed. Other motions are more sensitive to the FRI placement. For these motions we optimize an FRI policy using the value iteration algorithm.

The first step in formulating the value iteration is to identify the dynamics of interest. The memory and time complexity of the value iteration algorithm is exponential in the dimension of the dynamics, thus we must be careful to identify a low dimensional dynamics. Fortunately, due to the motion constraints (Section 3.1), the dynamics of the character may be characerized solely in terms of evolution on the low dimensional zero dynamics set.

Let $\boldsymbol{x} = (\theta, \dot{\theta})^T$ be the state on the zero dynamics set. Then the closed loop dynamics of a phase-indexed controller may written as a hybrid system involving both continuous and impulsive stages:

continuous dynamics (if $\theta_{min} \leq \theta < \theta_{max}$):

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\theta} \\ \boldsymbol{f}(\boldsymbol{x}, \text{FRI}^*) \end{bmatrix} \tag{5.2}$$

$$\text{FRI}^* = \Pi(\boldsymbol{x}) \tag{5.3}$$

impulsive dynamics (if $\theta = \theta_{max}$):

$$\boldsymbol{x}^+ = \begin{bmatrix} \theta_{min} \\ \boldsymbol{L}(\boldsymbol{x}^-) \end{bmatrix} \tag{5.4}$$

The impulsive dynamics are in the form of a transport function which instantaneously resets the phase-variable between symmetric left and right swing phases.

As can be seen from the dynamics, the only control input to the system is the placement of the FRI. The FRI represents the sole remaining degree of freedom in the control after accounting for feedback on the motion constraints. However, the allowable values of the FRI are limited to the base of support, reflecting the underactuated nature of the biped control problem.

As previously defined (Section 3.3.1) , the FRI policy is a function,

$$\Pi(\theta, \dot{\theta}) : \mathbb{S} \rightarrow [\text{FRI}^{x-}, \text{FRI}^{x+}], \tag{5.5}$$

which maps the zero dynamics state, $(\theta, \dot{\theta}) \in \mathbb{S}$ to an FRI value in the range $[\text{FRI}^{x-}, \text{FRI}^{x+}]$. The limits on the range of the FRI value depend upon the stage of the motion. However the stage of the motion is defined by the value of the θ variable, thus the limits can be known at any point on the zero dynamics. It is the goal of the policy optimization to determine the best use of this limited control.

The dynamics described so far are for the continuous state variables $(\theta, \dot{\theta})$, however, the

value iteration algorithm requires discrete states. Thus the next step is to approximate the continuous dynamics with a discrete one. Let $(\theta, \dot{\theta}) \in [\theta_{min}, \theta_{max}] \times [0, \dot{\theta}_{max}]$ and FRI $\in$ [FRI$^-$FRI$^+$] be the valid range of continuous states and actions. Then, let $\mathbb{S}$ and $\mathbb{A}$ be sets of discrete states and actions corresponding to samples on a grid in these ranges:

$$\mathbb{S} = \{s_{ij} | 0 \leq i < m, 0 \leq j < n, \} \tag{5.6}$$

$$s_{ij} = (\theta_i, \dot{\theta}_j) = (\theta_{min} + i\frac{\theta_{max} - \theta_{min}}{m-1}, j\frac{\dot{\theta}_{max}}{n-1}), \tag{5.7}$$

and

$$\mathbb{A} = \{a_k | 0 \leq k < p\} \tag{5.8}$$

$$a_k = \text{FRI}^- + i(\frac{\text{FRI}^+ - \text{FRI}^-}{p-1}). \tag{5.9}$$

Using these discrete sets, a probabilistic transition function is defined,

$$\mathbf{T}(s_{ij}, a_k) : \mathbb{S} \times \mathbb{A} \rightarrow (\mathbb{S} \times \mathbb{S} \times [0,1]) \cup \{\emptyset\}, \tag{5.10}$$

that maps a discrete state and action pair (s_{ij}, a_k) either to the failure state, $\emptyset$, or to a pair of adjacent states and a real number representing the probability of being in one or the other.

$\mathbf{T}$ is defined in terms of the continuous dynamics. Starting from discrete state s_{ij} and holding the FRI at a_k, the simulation is forward integrated until $\theta = \theta_{i+1}$. After integration, the simulation will be in the continuous state $(\theta_{i+1}, \dot{\theta}^f)$ and the transition function will be assigned a value,

$$\mathbf{T}(s_{ij}, a_k) = \{s_{i+1,\ell}, s_{i+1,\ell+1}, w\} \tag{5.11}$$

$$w = \frac{\dot{\theta}^f - \dot{\theta}_\ell}{\dot{\theta}_{\ell+1} - \dot{\theta}_\ell}$$

$$(\text{where } \dot{\theta}_\ell \leq \dot{\theta}^f < \dot{\theta}_{\ell+1}), \tag{5.12}$$

corresponding to a probability of being in either of the two nearest discrete states ($s_{i+1,\ell}$ or $s_{i+1,\ell+1}$). However, if at any time during forward integration the simulation leaves the valid range, or the motion constraints are violated by more than a given amount, the result of the transition function is marked as the failure state. The transition function is sampled in this manner for all combinations $(s_{ij}, a_k) \in \mathbb{S} \times \mathbb{A}$.

Once transitions have been calculated, the final step before performing value iteration is to define a value function.A simple value function which avoids the failure state and tracks the original motion speed is given by

$$\mathbb{V}(s_{ij}) = \begin{cases} -\infty & (\text{if } \mathbf{T}(s_{ij}, a) = \emptyset) \\ -c(s_{ij}) + \alpha[(1.0 - w) * \mathbb{V}(s_a) + w * \mathbb{V}(s_b)], & (\text{if } \mathbf{T}(s_{ij}, a) = \{s_a, s_b, w\}) \end{cases} \tag{5.13}$$

$$c(s_{ij}) = -k_2 e^{-k_1(\dot{\theta}_j - \hat{\dot{\theta}})^2} \tag{5.14}$$

$$a = \Pi(s_{ij}), \tag{5.15}$$

where $c(s_{ij})$ is a instantaneous cost that depends on whether the state is a failure state and $\hat{\dot{\theta}}$ is a reference value from the input motion.

Finally, the value iteration algorithm solves iteratively for the value function and policy that satisfy the discrete Bellman equation.

FRI Policy for Jumping

As an example of using phase-indexed tracking to design a controller for a motion other than walking, we constructed a jumping controller (see Section 3.4.3). We found that manually designed or feedforward strategies for controlling the FRI during the INIT and LAND stages of the controller fail to adequately regulate the forward speed of the jumping motion. In this section will describe how an FRI policy is optimized for jumping by attempting to regulate the desired value of $\dot{\theta}$ at the beginning of the TOEOFF stage.

The FRI policy is learned in the two dimensional discrete state space $\mathbb{S}$ containing states

$s_{ij} = (\theta_i, \dot{\theta}_j)$. The discrete states correspond to continuous states discretized into a 100×100 grid for a total of 10,000 discrete states. The actions space, $\mathbb{A}$, is also divided into 20 values of the FRI in the range $[\text{FRI}^{x-}, \text{FRI}^{x+}]$.

A discrete transition function is defined,

$$\boldsymbol{T} : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S} \cup \emptyset, \tag{5.16}$$

where $\emptyset$ is a failure state. The transition function is sampled by initializing the system to each discrete state, $s_{ij} \in \mathbb{S}$, and simulating with a fixed FRI, $a \in \mathbb{A}$, until $\theta = \theta_{i+1}$. If during simulation, the motion constraints are violated by more than a specified amount or the system state leaves the bounds of the discretized state space, the simulation is terminated and a transition to the failure state is recorded.

A value function is defined,

$$\mathbb{V}(s_{ij}) = \begin{cases} \|\dot{\theta}_i - \dot{\theta}_d\| & (\text{if } \theta_i = \theta_e) \\ \mathbb{V}(\mathbf{T}(\Pi(s_{ij})), & (otherwise), \end{cases} \tag{5.17}$$

(5.18)

where θ_e is the value of θ when the controller switches to the TOEOFF stage, $\dot{\theta}_d$ is the desired speed at this stage transition, and $\Pi : \mathbb{S} \rightarrow \mathbb{A}$ is the FRI policy. A value iteration then determines the value function and policy.

5.2.2 Discrete Stepping Policies

The purpose of a discrete stepping policy is to choose an appropriate sequence of steps to achieve a given goal. We have experimented with two types of discrete stepping policies.The first type of stepping policy chooses steps to successfully navigate a ground with constraints on where the character can step, to reach a desired goal position. The second type of stepping policy chooses steps in order to transition to a gait pattern that is interactively specified by the user. These are non-trivial tasks since not all gait patterns can be

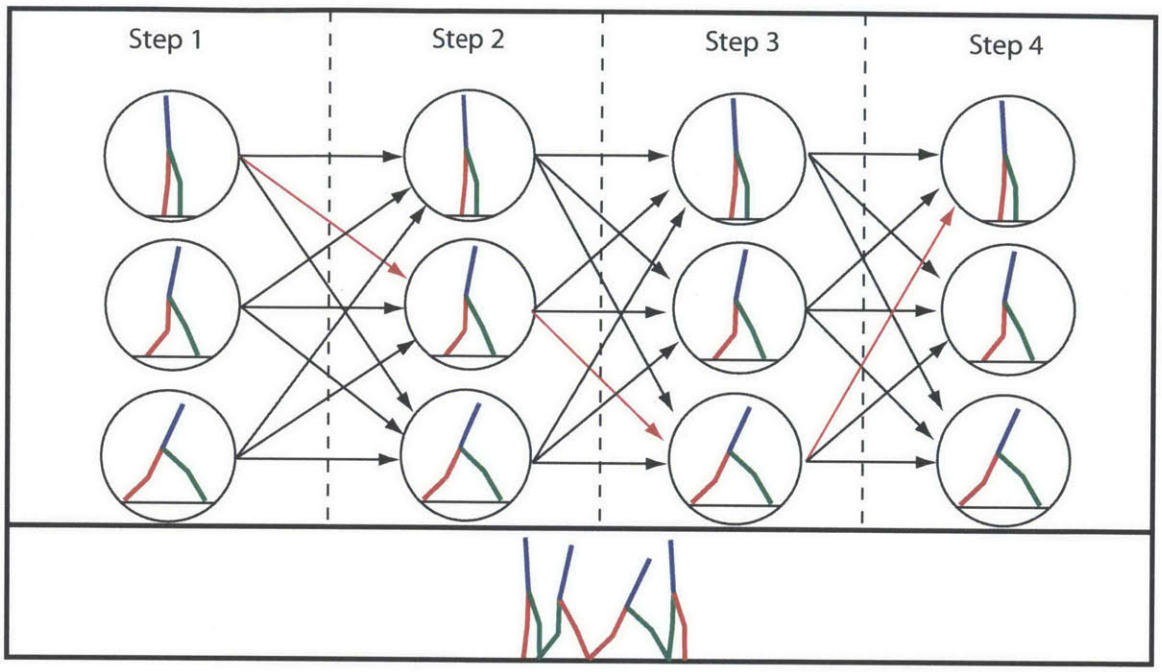

Figure 5-1: *Depicted is the operation of the constrained stepping controller for a walk consisting of four steps (at bottom). Arrows represent a fixed choice of motion constraints over the duration of the next step. Circles represented transition regions between steps. An optimized stepping policy chooses the best sequence of motion constraints (red arrows) to navigate a constrained terrain. The figure depicts only 3 possible step types, but we have successfully sequenced motions with 100 possible step types.*

directly transitioned between. The success of a transition depends upon the type of the gait as well as the rate of the $\dot{\theta}$ variable at the transition.

The desired behavior for both types of stepping policies may be described in terms of minimizing a cost function that assigns infinite value to states where the controller fails and assigns a large negative value to states that reach their goal. Failure states include falling down, failing to make forward progress, or entering a state where the motion constraints becomes unstable (i.e. computed joint torques become large or friction constraints are consistently violated).

Constrained Ground Navigation Stepping Policy

The goal of the ground navigation stepping policy is to choose a sequence of step transitions so that the character successfully navigates a terrain without stepping in regions designated

as holes. To construct the policy, an initial set of 100 motion constraints, $m_k \in \mathbb{M}$, is created using trajectory optimization. $\mathbb{M}$ consists of 10 different stride-lengths and 10 different speeds sampled on a grid at even intervals. An auxiliary set of motion constraints, $n_{k\ell} \in \mathbb{N} = \mathbb{M} \times \mathbb{M}$, which transition between pairs of the original steps, are also defined. These motion constraints are implicitly constructed by assuming a smooth-in, smooth-out blend of the spline parameters between each pair of the original constraints. This results in 100,000 possible step controllers, or 100 viable transitions from each of the original 100 steps (see Figure 5-1). Although not all the blends result in feasible motion constraints, transitions between steps with similar stride length and speed usually succeed for some range of initial velocity. The bad transitions are pruned automatically when sampling the transition function for the policy optimization.

The policy optimization is constructed in a 3-dimensional state space with discrete values $s_{ijk} = (\dot{\theta}_i, p_j, m_k) \in \mathbb{S}$. The first dimension is the discretized values of the initial state velocity, $\dot{\theta} \in [0, \dot{\theta}_{max}]$, the second dimension is the discretized locations of the stance foot on the ground plane, $p \in [0, p_{max}]$, and the third dimension is the current motion state, $m_k \in \mathbb{M}$. In the policies we optimized, the set has size $20 \times 500 \times 100$.

The goal of the value iteration is to optimize a policy function,

$$\Pi(s_{ijk}) : \mathbb{S} \rightarrow \mathbb{A}, \tag{5.19}$$

which maps discrete states to actions. In this case, actions, $a_\ell \in \mathbb{A} = \mathbb{M}$, are drawn from the same set as the original motion since we start by assuming that all transitions are feasible (including self transitions).

A transitions function, $\mathbf{T} : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S} \cup \emptyset \cup \gamma$, is defined which maps state and action pairs to a new states or, possibly, to the special failure or goal state, $\emptyset$ or γ. The value of $T(s_{ijk}, a_\ell)$ is computed by initializing the simulation to the continuous state equivalent of the discrete zero dynamics state $(\theta_0, \dot{\theta}_i)$ and then performing the motion constraint blend that transitions from step m_k to m_ℓ. Transitions to states where the character's foot is on a gap map to the failure state and transitions to the goal location, $p_j = p_{goal}$, map to a success state. If

at any time during forward simulation the motion constraints diverge from zero by more than a specified amount or the character stops making forward progress (i.e., $\dot{\theta} < 0$), the simulation is stopped and the transition is mapped to the failure state. Note that simulated steps are identical modulo the ground location, p_j, thus simulation results may be reused when computing the transition function at each ground location.

A value function is defined,

$$\mathbb{V}(s_{ij}) = \begin{cases} \beta & (\text{if } p_j = p_{goal}) \\ -c(s_{ij}) + \alpha \mathbb{V}(r), & (\text{otherwise}) \end{cases} \tag{5.20}$$

$$r = \mathbf{T}(\Pi(s_{ij})), \tag{5.21}$$

where β is a large positive reward for reaching the goal state and $c(s_{ij})$ is a per-step cost that can be used to change how the controller reaches the goal state. For example, if the per-step cost is proportional to the length of the step, s_{ij}, then the resulting value function will penalize taking long steps and prefer a strategy that reaches the goal using shorter steps. Similarly, if the per-step cost is proportional to the step duration, the resulting policy will try to reach the goal as quickly as possible.

User-Guided Stepping Policy

The user-guided stepping policy switches between a connected graph of step transitions that exhibit variation along the dimensions for which we were able to edit the stepping motions using the techniques of Chapter 4. The policy is able to choose motion transitions that switch from the current step type to the user specified step type in a minimal number of transitions. The user can interactively specify both the desired gait speed and desired step length and the controller will determine a sequence of transitions to achieve a steady state gait with those parameters.

The first step in constructing the user-guided stepping policy is to create an action set. The set is to produced by creating edited versions of a single stepping motion along two dimension: step length and step speed. In the policies we created, we sampled a grid of 3 step

lengths and 3 step speeds for a total of 9 step types, $m_j \in \mathbb{M}$. A set of transition motion constraints (Section 4.1.4) are created between each step type for a total of 81 unique motion constraints, $n_\ell \in \mathbb{M} \times \mathbb{M}$. The trajectory optimization for some of these transitions may fail to converge or may produce motions that look unnatural. After manually examining the results of the trajectory optimization and tuning the objective weights, the motions which still fail to converge or look too unnatural are pruned from the set. The resulting action set, $n_\ell \in \mathbb{N} \subset \mathbb{M} \times \mathbb{M}$, represents a partially connected graph of possible motion transitions. As would be expected, transitions between motions with similar parameters tend to succeed and transitions between motions that differ in more than one dimension tend to fail.

Next, an FRI policy, Π^ℓ, is optimized independently for each of the successful step transitions and a discrete step transition function,

$$\mathbf{T}(s_{ij}, a_k) : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S} \cup \emptyset, \tag{5.22}$$

is sampled, with $s_{ij} = (\dot{\theta}_i, m_j) \in \mathbb{S}$ and $a_k \in \mathbb{A} = \mathbb{M}$. The process of sampling the transition function is similar to the one described for ground navigation but without a state dimension corresponding to the location on the ground plane. Using the sampled transition function, 9 policies, Π_{m_k}, are created, each aiming to bring the controller into a steady state gait using motion constraints m_k.

A value function for each policy is defined which takes into account the individual FRI policies. For policy Π_{m_k}, the value function takes the form:

$$\mathbb{V}(s_{ij}) = \begin{cases} -\infty & (\text{if } s_{ij} = \emptyset) \\ -c(s_{ij}) + \beta + \alpha \mathbb{V}(r), & (\text{if } r = m_k), \\ -c(s_{ij}) + \alpha \mathbb{V}(r), & (\text{otherwise}) \end{cases} \tag{5.23}$$

$$c(s_{ij}) = -\Pi^\ell(\theta_0, \dot{\theta}_i), \tag{5.24}$$

$$r = \mathbf{T}(\Pi_{m_k}(s_{ij})), \tag{5.25}$$

where β is a positive constant that rewards transitions to the desired motion constraints.

5.3 Results

5.3.1 FRI Policies for Walking

We have designed FRI policies for three different styles of walking as well as several adaptations of these walks, including variations on the step length, step speed and ground incline. All three gaits failed to function well using heuristically designed FRI policies we tried. For these gaits, heuristically designed FRI policies exhibited large fluctuations in the speed profile and often resulted in failure. On the other hand, the optimized policies always succeeded.

The only walking gaits that functioned well without a optimized FRI policy were those gaits with a much simpler, flat footed stepping style. The gaits that required an optimized the FRI policy involved a life-like heel-strike and toe-off phase. These gaits all exhibit a brief moment before heel-strike when the only contact between the character and the ground is a point on the stance toe. During this stage of the motion, the FRI is restricted to a point and the phase-indexed controller has no control over speed. It is likely that this brief moment of underactuation contributed to the poor performance of manually designed FRI policies.

We found that an optimized FRI policy was especially important for stepping variations that changed the incline of the ground plane. This is because the FRI is directly related to insertion or removal of momentum necessary to ascend or descend a slope.

The time it took to optimize an FRI policy was dominated by the time it took to sample the transition function. For all gaits we chose a state and action discretization with $n = 100$ values of θ, $m = 100$ values of $\dot{\theta}$, and $p = 10$ values of the FRI. So sampling time was roughly proportional to the time it takes to simulate $m * p$ full steps. In practice, this took around 5 to 10 minutes on a desktop computer.

The value iteration itself occurred much faster. One reason for this is that despite the large number of states, the transition function is highly structured. The transition function will always transport the state forward along the phase variable θ. Our value iteration can converge significantly faster than one with an unstructured transition function since

on each iteration of the algorithm we can always start with the greatest value of θ and move backwards. This ensures that on iteration, i, of the value iteration we are computing the exact optimal i-step finite horizon policy for the discrete system. In practice, with a discount parameter $\alpha = 0.999$, we found that the infinite horizon value function sufficiently converges (so as to no longer significantly affect the policy) after 1000 iterations. This process runs to convergence in under half a second.

Figure 5-2 depicts a typical optimized FRI policy and the associated value function. One consistent feature among all the optimized polices was that they exhibited a quick switch between extreme values of the FRI along a border in state space.

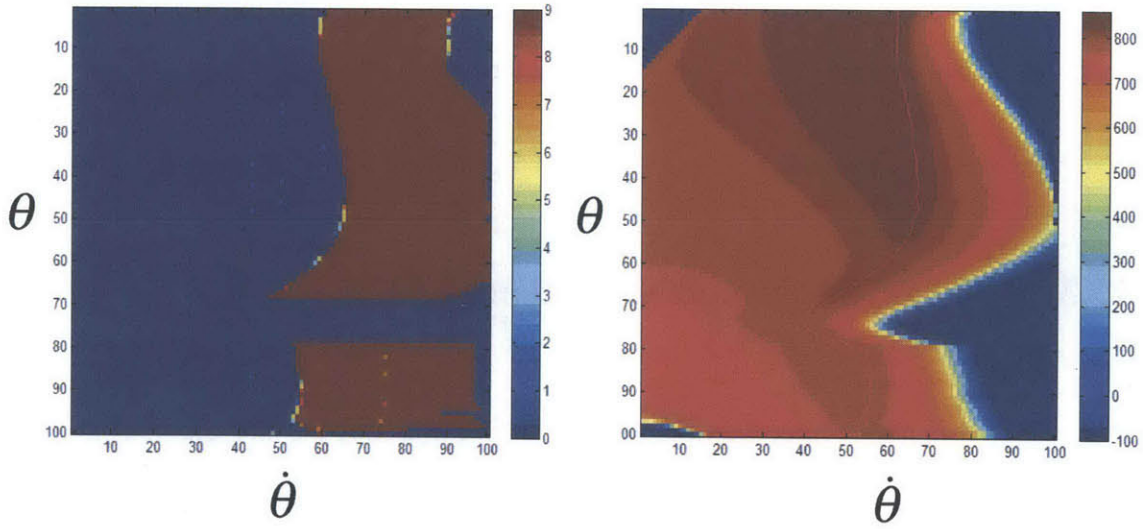

Figure 5-2: *An optimized FRI policy (left) and value function (right) for a typical forward walking controller using a cost function that rewards similarity to the speed profile of the original motion (Equation 5.13). The axis are labelled according to the discrete sample index of the θ_i and $\dot{\theta}_j$ variable along a regular grid. The color in the policy represents the discrete index of the FRI action. The red curve on the cost function is the trajectory of the original motion. The value is highest near the original motion. The policy exhibits a "bang bang" style of control where the FRI switches quickly from one extreme to the other at a boundary in state space. The horizontal band in the policy occurs during a stage of the motion where the FRI is restricted to a point, so all discrete actions are equivalent.*

5.3.2 FRI Policy for Jumping

With a constant FRI policy the jump controller either speeds up or slows down until failure. This can be seen by examining a return map of the $\dot{\theta}$ variable (see Figure 5-3). The FRI policy reshapes the return map, indicating stability of the jump cycle. The jump controller with the learned policy is able to jump indefinitely on flat ground and even up slight inclines.

Even with the FRI policy the controller is unable to recover when $\dot{\theta}$ is too small or too large. When $\dot{\theta}$ is too small, the character does not generate sufficient momentum to carry the body forward and falls backwards leading to immediate failure. When $\dot{\theta}$ is too large, the the character is unable to generate sufficient braking power to decrease the forward momentum of the previous jump. The controller will continue to function for a couple jumps, increasing speed on each one, but will eventually fall forward and fail.

5.3.3 Constrained Ground Navigation Stepping Policy

The main cost in performing the value iteration is in computing the discrete transition function, as this requires performing a number of simulations equal to the number of discrete states times the number of discrete actions. But since the steps are identical modulo horizontal translation on the ground plane, the number of simulation that need to be performed is reduced by a factor of 500. Once the transition function is tabulated, it is relatively quick to compute an optimized policy. The policy can be adapted for a different sequence of gaps or a different goal location without having to resample the transition function. Additionally, our action set is rich enough that it is possible to include a secondary criteria. We demonstrated this by designing stepping policies that preferred to take more or fewer steps to reach the goal location. These variations on the policy can also be computed without resampling the transition function (see Figure 5-4).

5.3.4 User-Guided Stepping Policy

We produced a discrete stepping policy for a character with feet. This policy switches between a connected graph of step controllers that exhibits variation along the dimensions for which we were able to edit the motion using the trajectory optimization of Chapter 4. The user can interactively specify both the desired gait speed and desired step length and the controller will determine a sequence of transitions to achieve a steady state gait with those parameters (see Figure 5-5).

5.4 Discussion

In this chapter we have presented four different ways in which value iteration can be used in conjunction with a phase-indexed tracking controller. It is important to note that although value iteration aims to produce globally optimal control strategies, the resulting controllers are only approximately optimal. The FRI policies are approximate due to the fact that the discrete dynamics only approximates the continuous dynamics. Both the FRI and stepping polices are approximate due to the fact that transition functions are only accurate if the motion constraints are precisely satisfied ($y = 0$). When a controller is perturbed, we can think of recovery as happening in a two stage process. First, the motion constraints converge toward zero during which time we have no guarantee that the optimized policy is behaving well. However, once on the zero dynamics set, the character behaves in an approximately optimal manner using the optimized policy.

A fundamental challenge we have address is the need to incorporate predictions into policy design. Prediction is necessary whenever a greedy policy is bound to fail, which is usually the case for bipeds due to their underactuation. In regards to prediction, we have shown that by restricting the space of motions (via motion constraints) it becomes tractable to sample transition functions and use them to optimize policies. Of course the use of motions constrains in this manner is a double edged sword. While restricting the space of motions yields easier prediction, it also restricts the set of policies that can be developed. This

restriction will ultimately result in policies that are suboptimal with respect to the full capabilities of a character model. A prime example of this is the inability of a phase-indexed controllers to intentionally use an inverted pendulum-like hip strategy to regain dynamics balance. Swinging the leg forward or the torso backwards to generate an angular moment would violate the motions constraints. In general, the lack of any consideration for what happens when the characters diverge from the motion constraints is problematic from a theoretical perspective, but in practice, these problems are somewhat mitigated by the fact that the form of the motion constraints used are surprisingly stable (although no formal guarantee of this is provided).

Another important issue we have touched on in this chapter is the idea of sampling a simulation. Although sampling of transition function through numerical means might be considered a less elegant solution than deriving an analytical policy, it offers one clear advantage. It allows the policies to adapt to the precise dynamics of the simulator. Simulators used for computer graphics applications often trade accuracy for speed resulting in approximations of physics that are hard to identify. Even when these approximations are known, they often involve complex numerical algorithms that do not yield easily differentiable forward dynamics functions. This can make the rectification of motions a difficult and slow process. One possible use of our approach is as a two-step rectification process. First, trajectory optimization is used to find motions which are approximately feasible using gradient-based optimization for speed. Second, a low-dimensional adaptation (the FRI position) is made using direct sampling of the simulation to ensure true motion feasibility. We have shown that using this idea an input motion trajectory can be turned into a fully functioning control strategy within minutes. This should be contrasted with other high dimensional sampling strategies (e.g., [63, 66]) that require optimization times on the order of hours or days to produce stylistically pleasing motions in simulation.

5.5 Conclusion

In this chapter, we have discussed the use of policy optimization to design better control policies for phase-indexed controllers. We have shown how to optimize a policy for placement of the FRI point in order to regulate the walking speed of gaits and we have shown how to optimize discrete step transition policies that navigate terrain as well as respond to user input. In doing so we have touched on several important challenges related to the design of controllers for simulated characters.

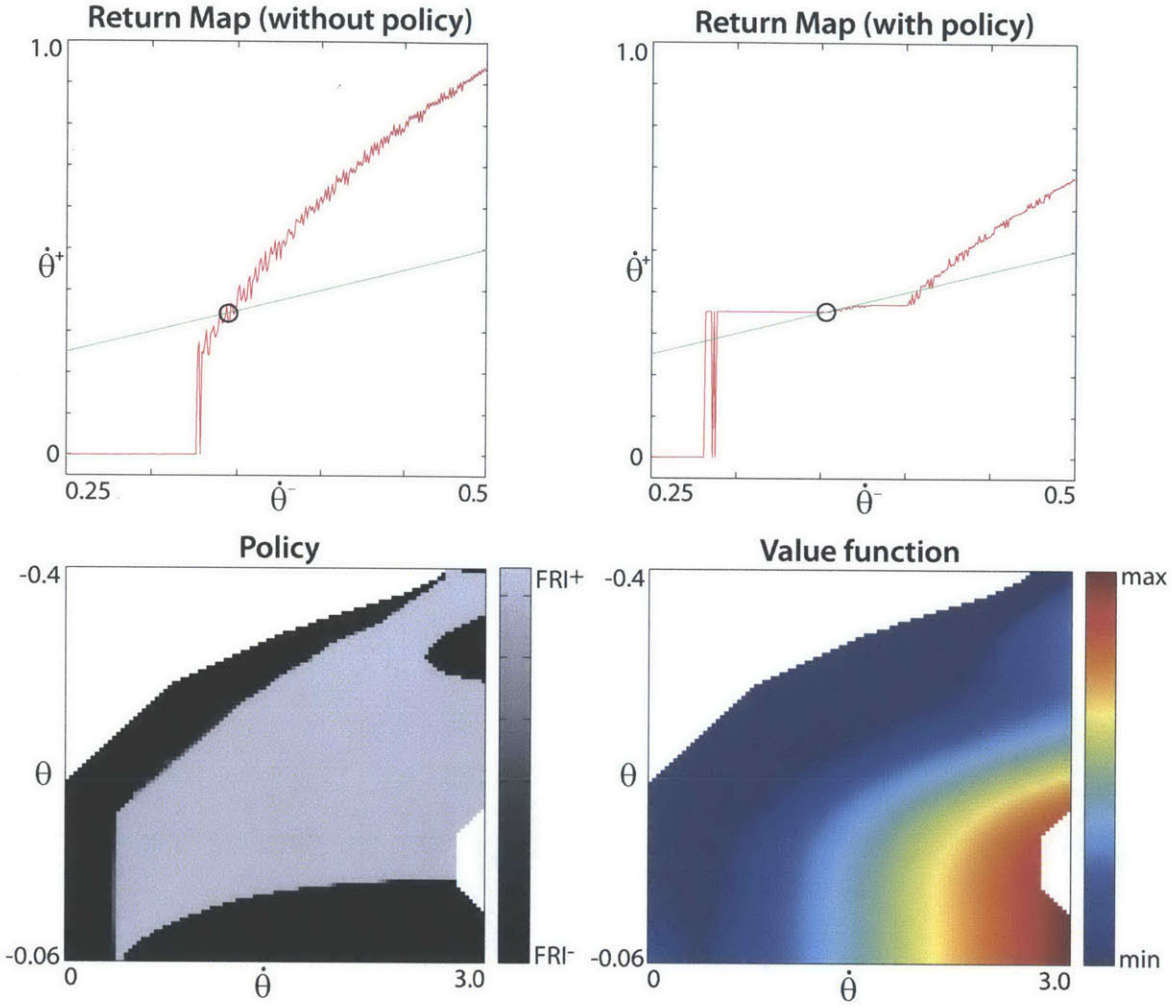

Figure 5-3: *A return map is an indicator of stability. It maps the value of $\dot{\theta}$ just prior to the INIT stage of one jump to the value of $\dot{\theta}$ at the same point in the next jump in the sequence. If the slope of the return map (red line) is less than 1 (green line) at the point of intersection (circle), then successive jumps will converge toward the intersection and the controller will be stable. Otherwise the controller will speed up or slow down until failure. We learn an FRI policy (left-bottom) that results in a stable return map (top-right). Although we allow for a range of FRI values, the controller chooses to rapidly switch between extreme values of the FRI. In the policy and value function, white regions correspond to states from which the controller will fail. The value function predicts, given the current state $(\theta, \dot{\theta})$, how close the controller will be to the desired value of $\dot{\theta}$ at the beginning of the next jump.*

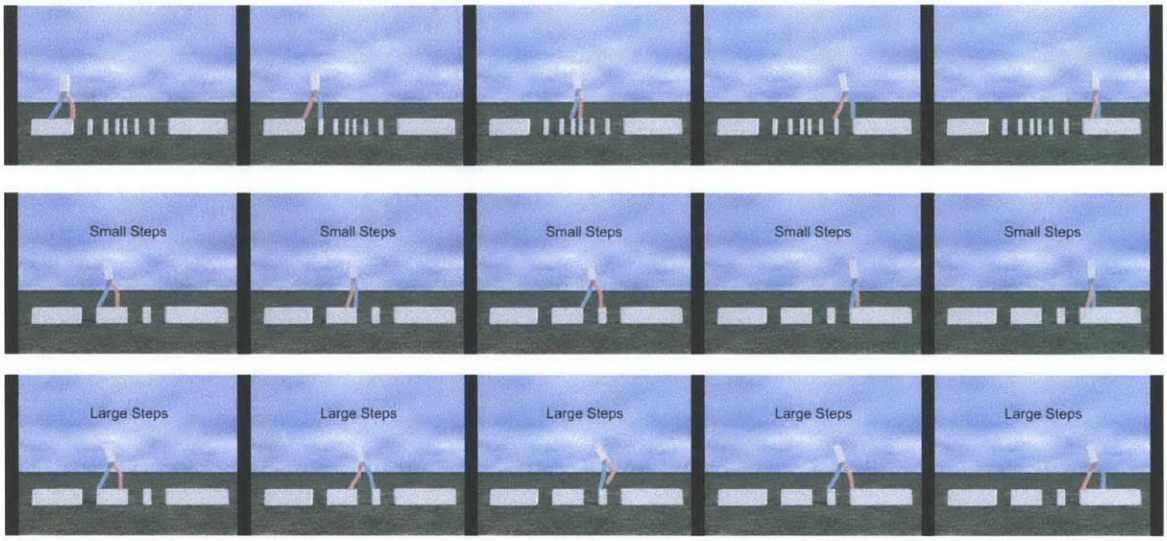

Figure 5-4: *The figure depicts chronologically ordered (left to right) still frames from simulations using constrained ground navigation stepping policies. The bottom two rows shows navigation over the same terrain using different value functions that reward taking either shorter (second row) or longer (third row) steps. Note the difference in body angle as the controller dynamically prepares to take a larger step (third row, second column).*

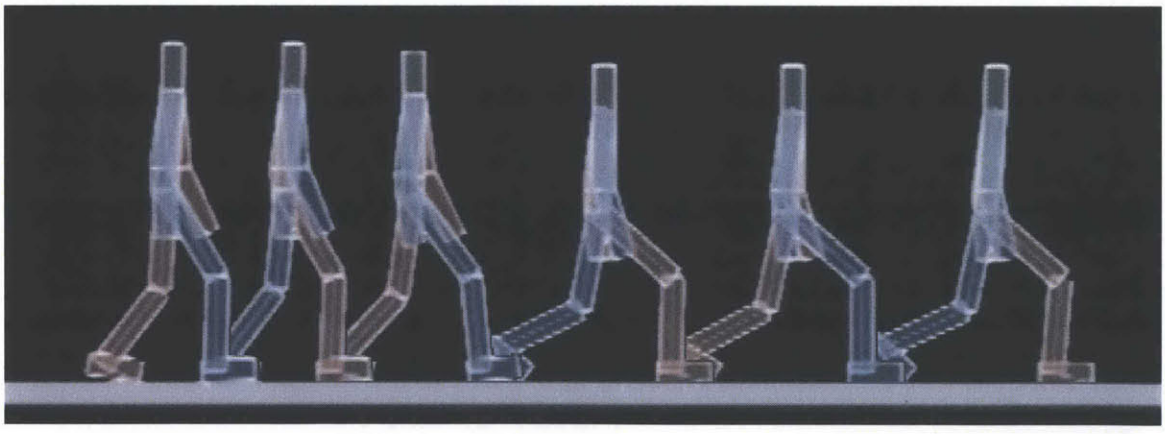

Figure 5-5: *The figure depicts a step sequence (left to right) of the user-guided stepping policy after a user interactively requests that the controller transition to taking large, fast steps. The controller selects a sequence of transition by first taking a fast step of the same length, then a middle length step, and then finally steps of the desired length and speed.*

Chapter 6

Conclusion

We have introduced the concepts of phase-indexed tracking and policy optimization in a reduced dimensional subspace through the use of motion constraints. Phase-indexed tracking robustly emulates an input motion while allowing flexibility in the timing. 2D walking controllers designed in this manner are more robust and can withstand larger force disturbances using less control effort when compared to time-indexed controllers, such as the recent non-linear quadratic regulator [45].

Low-dimensional policy optimization addresses the need to incorporate prediction into the design of control policies. We have demonstrated continuous policies for deciding the placement of the FRI point in order to regulate the speed of walking and to stabilize a sequence of broad jumps. We have also demonstrated discrete policies for choosing optimal steps to guide a walking character over a constrained terrain and to respond to interactive user input. In the future we envision using optimal policy learning to model the high-level intentions of simulated characters in a similar fashion to how it has been used in kinematic motion controllers [60]. Adding physical considerations to these controllers would improve their versatility.

The robustness of our walking controller is comparable to the best manually-designed feedback laws, such as the recent SIMBICON controller [67]. However, it is interesting to note that the two controllers use contrasting strategies to achieve robustness. The SIMBICON

controller changes the width of each step based upon a feedback loop whereas our controller only changes the timing. A hybrid strategy could improve both schemes.

6.1 Future Work

A major avenue of future investigation is 3D. Currently our controller have not been extended to the 3D because 3D motions are inherently less stable and require more complex feedback schemes. Motion constraints have been used in 3D to generate simplified walkers with point feet [13], but that approach did not result in controllers that are capable of tracking motion data. The next step is to refine this (or similar) ideas to develop 3D tracking controllers.

One of the main motivations of the work presented in this thesis was to investigate ways in which controllers could be designed quickly and simply by non-control experts. Although the methods used in this thesis aim to be automatic, general, and fast, very little has been done in the way of validating these assumptions on a larger dataset or with non-experts users. A definite avenue for future work is to design a user interface and workflow that would allow for further refinement of the proposed methods.

The types of motions explored in this thesis barely scratch the surface of the full range of behaviors exhibited by real humans. Future works must investigate control strategies for multiple characters interacting with each other or for characters manipulating objects. Beyond humans, there is the challenge of simulating and controlling the motions of the entire animal kingdom. The possibilities for simulation and control are as vast as the real world itself. There is certainly more work to be done.

Bibliography

[1] Yeuhi Abe, Marco da Silva, and Jovan Popović. Multiobjective control with frictional contacts. In *Symposium on Computer Animation (SCA)*, pages 249–258, 2007.

[2] Christopher G. Atkeson and Jun Morimoto. Nonparametric representation of policies and value functions: A trajectory-based approach. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 1611–1618. MIT Press, Cambridge, MA, 2002.

[3] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 23–34, New York, NY, USA, 1994. ACM.

[4] David Baraff. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1996. ACM.

[5] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

[6] H. Benbrahim and J. Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22:283ï£¡302, 1997.

[7] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996.

[8] John T. Betts. *Practical methods for optimal control using nonlinear programming*, volume 3 of *Advances in Design and Control*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[9] Katie Byl and Russ Tedrake. Approximate optimal control of the compass gait on rough terrain. In *Proceedings International Conference on Robotics and Automation*, 2008.

[10] C. Byrnes and A. Isidori. Asymptotic stabilization of minimum phase nonlinear systems. *IEEE Transactions on Automatic Control*, 36:1122–1137, October 1991.

[11] C. Canudas. On the concept of virtual constraints as a tool for walking robot control and balancing. *Annual Review in Control*, 28:157–166, 2004.

[12] C. Chevallereau, D. Djoudi, and J. W. Grizzle. Stable bipedal walking with foot rotation through direct regulation of the zero moment point. *IEEE Transactions on Robotics*, 2008.

[13] SChristine Chevallereau, J. W. Grizzle, and Ching-Long Shih. Asymptotically stable walking of a five-link underactuated 3d bipedal robot. *IEEE Transactions on Robotics*, 2008.

[14] C. Chew and G. A. Pratt. Dynamic bipedal walking assisted by learning. *Robotica*, 20:477–491, 2002.

[15] Michael B. Cline and Dinesh K. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14-19, 2003, Taipei, Taiwan*, pages 3744–3751. IEEE, 2003.

[16] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. In *International Conference on Robotics and Automation (ICRA)*, volume 1, pages 228–233. IEEE, 1988.

[17] Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Robust task-based control policies for physics-based characters. *ACM Trans. Graph.*, 28(5):1–9, 2009.

[18] Stelian Coros, Philippe Beaudoin, KangKang Yin, and Michiel van de Panne. Synthesis of constrained walking skills. *ACM Trans. Graph. (Proc. Siggraph Asia)*, XX(X), 2008.

[19] Richard Cottle, Jong-Shi Pang, and Richard E. Stone. *The linear complementarity problem*. Academic Press, 1992.

[20] Marco da Silva, Yeuhi Abe, and Jovan Popović. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics*, 27(3):82:1–82:10, August 2008.

[21] C. De Boor. *A Practical Guide to Splines*. Number v. 27 in Applied Mathematical Sciences. Springer-Verlag, 1978.

[22] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot. *Int. J. Rob. Res.*, 27(2):213–228, 2008.

[23] Tom Erez and William D. Smart. Bipedal walking on rough terrain using manifold control. pages 1539–1544, 2007.

[24] Kenny Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2):12, 2007.

[25] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, pages 251–260, August 2001.

[26] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Trans. Graph.*, 22:417–426, July 2003.

[27] Roy Featherstone. *Rigid Body Dynamics Algorithms.* Springer Science+Business Media, 2008.

[28] Philip E. Gill, Walter Murray, Michael, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization, 1997.

[29] Ambarish Goswami. Foot rotation indicator (fri) point: A new gait planning tool to evaluate postural stability of biped robots. In *IEEE International Conference on Robotics and Automation*, pages 47–52, 1999.

[30] H. Hemami, K. Barin, and Y.-C. Pai. Quantitative analysis of the ankle strategy under platform disturbance. *IEEE Transactions on Neural Systems and Rehabilitation*, 14:470–80, 1996.

[31] A. Isidori and C. H. Moog. On the nonlinear equivalent of the notion of transmission zeroes. In *Proc. of the IIASA Conference: Modeling and Adaptive Control*, pages 146–57. Springer-Verla, 1988.

[32] D.H. Jacobson and D.Q. Mayne. *Differential dynamic programming*. Modern analytic and computational methods in science and mathematics. American Elsevier Pub. Co., 1970.

[33] Ollie Johnston and Frank Thomas. *The illusion of Life, Disney Animation.* Hyperion, New York, 1981.

[34] R. E. Kalman. Contributions to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana*, 5:102–119, 1960.

[35] Andrew Witkinand 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.

[36] J. Zico Kolter, Adam Coates, Andrew Y. Ng, Yi Gu, and Charles DuHadway. Space-indexed dynamic programming: learning to follow trajectories. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 488–495, New York, NY, USA, 2008. ACM.

[37] Joseph F. Laszlo, Michiel Van De Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. In *In Proceedings of ACM SIGGRAPH*, pages 155–162. Press, 1996.

[38] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[39] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph.*, 24(3):1071–1081, July 2005.

[40] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Trans. Graph.*, 21:408–416, July 2002.

[41] Adriano Macchietto, Victor Zordan, and Christian R. Shelton. Momentum control for balance. pages 1–8, 2009.

[42] Ian Manchester, Uwe Mettin, Fumiya Iida, and Russ Tedrake. Stable dynamic walking over rough terrain. In CÃl'dric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, volume 70 of *Springer Tracts in Advanced Robotics*, pages 123–138. Springer Berlin / Heidelberg, 2011.

[43] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, pages 181–ff., New York, NY, USA, 1995. ACM.

[44] Jun Morimoto, Jun Nakanishi, Gen Endo, Gordon Cheng, Chris Atkeson, , and Garth Zeglin. Poincare-map-based reinforcement learning for biped walking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*, April 2005.

[45] Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph.*, 28(3):1–9, 2009.

[46] H. Nijmeijer and A. J. van der Schaft. *Nonlinear Dynamical Control*. Springer-Verlag, 1989.

[47] Marc H. Raibert. Legged robots that balance. 1986.

[48] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. 25(4):349–358, July 1991.

[49] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, August 2004.

[50] Dana Sharon and Michiel van de Panne. Learning to control physics-based stylized walking. In *IEEE International Conference on Robotics and Automation*, 2005.

[51] A. Shiriaev, J. W. Perram, and C. Canudas-de Wit. Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach. *IEEE Transactions on Automatic Control*, 50:1164–1176, 2005.

[52] Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics*, 26(3):107:1–107:9, July 2007.

[53] M W Spong. Partial feedback linearization of underactuated mechanical systems. *Proceedings of IEEERSJ International Conference on Intelligent Robots and Systems IROS94*, 1:314–321, 1994.

[54] R.F. Stengel. *Optimal Control and Estimation*. Dover Books on Advanced Mathematics. Dover Publications, 1994.

[55] David Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[56] Adnan Sulejmanpašić and Jovan Popović. Adaptation of performed ballistic motion. *ACM Trans. Graph.*, 24(1):165–179, January 2005.

[57] Yuval Tassa, Tom Erez, and William D. Smart. Receding horizon differential dynamic programming. *Advances in Neural Information Processing Systems*, 20:1465–1472, 2008.

[58] Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *Int. J. Rob. Res.*, 29(8):1038–1052, July 2010.

[59] Russ Tedrake and H. Sebastian Seung. Improved dynamic stability using reinforcement learning. In *5th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, pages 341–348, 2002.

[60] Adrien Treuille, Yongjoon Lee, and Zoran Popović. Near-optimal character animation with continuous control. *ACM Transactions on Graphics*, 26(3), July 2007.

[61] Michiel Van De Panne and Alexis Lamouret. Guided optimization for balanced locomotion. pages 165–177, 1995.

[62] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. *ACM Trans. Graph.*, 28:60:1–60:8, July 2009.

[63] J.M. Wang, D.J. Fleet, and A. Hertzmann. Optimizing walking controllers. *ACM Transactions on Graphics*, 28(3), 2009.

[64] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, Taylor & Francis Group, 2007.

[65] P. B. Wieber. On the stability of walking systems. In *International Workshop on Humanoid and Human Friendly Robotics*, 2002.

[66] KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics*, 27(3), 2008.

[67] Kangkang Yin, Kevin Loken, and Michiel van de Panne. SIMBICON: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):105:1–105:10, July 2007.

[68] 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.