Physics-based Character Locomotion Control with Large Simulation Time Steps

David Andrew Greer

October, 2015

Bournemouth University

A dissertation submitted in partial fulfillment of the
requirements for the degree of Engineering Doctorate

Abstract

Physical simulated locomotion allows rich and varied interactions with environments and other characters. However, control is difficult due to factors such as a typical character's numerous degrees of freedom and small stability region, discontinuous ground contacts, and indirect control over the centre of mass. Previous academic work has made significant progress in addressing these problems, but typically uses simulation time steps much smaller than those suitable for games. This project deals with developing control strategies using larger time steps. After describing some introductory work showing the difficulties of implementing a handcrafted controller with large physics time steps, three major areas of work are discussed.

The first area uses trajectory optimization to minimally alter reference motions to ensure physical validity, in order to improve simulated tracking. The approach builds on previous work which allows ground contacts to be modified as part of the optimization process, extending it to 3D problems. Incorporating contacts introduces difficult complementarity constraints, and an exact penalty method is shown here to improve solver robustness and performance compared to previous relaxation methods. Trajectory optimization is also used to modify reference motions to alter characteristics such as timing, stride length and heading direction, whilst maintaining physical validity, and to generate short transitions between existing motions.

The second area uses a sampling-based approach, previously demonstrated with small time steps, to formulate open loop control policies which reproduce reference motions. As a prerequisite, the reproducibility of simulation output from a common game physics engine, PhysX, is examined and conditions leading to highly reproducible behaviour are determined. For large time steps, sampling is shown to be susceptible to physical invalidities in the reference motion but, using physically optimized motions, is successfully applied at 60 time steps per second.

Finally, adaptations to an existing method using evolutionary algorithms to learn feedback policies are described. With large time steps, it is found to be necessary to use a dense feedback formulation and to introduce phase-dependence in order to obtain a successful controller, which is able to recover from impulses of several hundred Newtons applied for 0.1s. Additionally, it is shown that a recent machine learning approach based on support vector machines can identify whether disturbed character states will lead to failure, with high accuracy (99%) and with prediction times in the order of microseconds.

Together, the trajectory optimization, open loop control, and feedback developments allow successful control for a walking motion at 60 time steps per second, with control and simulation time of 0.62ms per time step. This means that it could plausibly be used within the demanding performance constraints of games. Furthermore, the availability of rapid failure prediction for the controller will allow more high level control strategies to be explored in future.

Acknowledgements

I would like to thank the Centre for Digital Entertainment, EPSRC and NaturalMotion for giving me the opportunity to work on a fascinating and challenging problem. My academic supervisors, Jian J Zhang and Zhidong Xiao, and industrial supervisors, Joss Knight and Alberto Aguado, provided invaluable guidance and encouragement, allowed me considerable latitude in choosing the project direction, and displayed great patience.

The whole tech team at NaturalMotion, and especially the research and behaviour groups, were tremendously helpful with explaining NaturalMotion's code and tools, and contributing many important insights on control strategies. I also made many good friends there.

The other research engineers at the CDE also provided many illuminating discussions, as well as some great memories and friendships. In particular, Chris Lewin wrote the Phozon software used to produce the ray-traced videos accompanying this thesis, and provided support with using it.

All the CDE project co-ordinators, and particularly Daniel Cox, were incredibly helpful with organization and support.

Contents

List of Figures

List of Tables

List of Abbreviations

Dynamics and Mechanics

COM	Centre of Mass
COP	Centre of Pressure
DOF	Degree of Freedom
EOM	Equation of Motion
GRF	Ground Reaction Force
IK	Inverse Kinematics
IP	Inverted Pendulum
IPC	Inverted Pendulum on Cart
IPM	Inverted Pendulum Model
LCP	Linear Complementarity Problem
SLIP	Spring Loaded Inverted Pendulum
ZMP	Zero Moment Point

Biological Control Terms

CNS	Central Nervous System
CPG	Central Pattern Generator
MTU	Musculotendon Unit

Control Terms

FSM	Finite State Machine
HJB	Hamilton-Jacobi-Bellman
LQG	Linear Quadratic Regulator with Gaussian Noise
LQR	Linear Quadratic Regulator
NQR	Nonlinear Quadratic Regulator
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative

Algorithms and Methods

CMA	Covariance Matrix Adaptation
FEL	Feedback Error Learning
GA	Genetic Algorithm
LDKL	Local Deep Kernel Learning
NN	Neural Network
PCA	Principal Component Analysis
PPCA	Probabilistic Principal Component Analysis
QP	Quadratic Programming

RBF	Radial Basis Function
SAN	Sensor-Actuator Network
SO / TO	Spacetime Optimization / Trajectory Optimization
SQP	Sequential Quadratic Programming
SVM	Support Vector Machine

Miscellaneous

CPU	Central Processing Unit
ODE	Open Dynamics Engine

Chapter 1

Introduction

1.1 Research Problem

Physics-based locomotion methods, which combine physically simulated characters with control strategies which mimic biological behaviour, have the potential to improve many aspects of character animation, including interactiveness, motion variety and naturalness. However, formulating effective controllers is a difficult task as characters have low instrinsic stability and many degrees of freedom to regulate, and ground contacts introduce problematic discontinuities. In designing controllers, it is also difficult to balance the competing objectives of robustness, motion style and naturalness, and controllability and responsiveness.

NaturalMotion's morpheme software is animation based (see Section 1.2, and the capability to interact physically with the environment is limited to one-way transfer of momentum from characters to the environment, and ragdoll simulations without sophisticated control. euphoria has many impressive physics-based behaviours, but does not have the functionality to reproduce a reference motion. Therefore it would be beneficial to develop methods which are able to physically track reference motions such as locomotion, under the same simulation conditions typically used in euphoria.

There are many existing examples of physics-based controllers, and Chapter 2 gives an overview of the large body of existing work, but often small simulation time steps are used and typically computational performance is not sufficient for interactive applications such as games, where animation will run alongside many other tasks. This doctoral project, in collaboration with NaturalMotion, draws on existing work but aims to investigate the extent to which the simulation conditions and computational requirements of physics-based control can be brought in line with the performance demands of games.

To illustrate the effect of simulation time step on a control problem, consider a simple toy example. A simple 2D rigid pendulum, with a control torque, τ, applied at the pivot in order to follow a desired trajectory for the pendulum angle θ, models some minimal elements of a locomotion control problem. The effect of the simulation time step was examined by applying the following control policy

$$\tau = k(\theta_d - \theta) + 2\sqrt{k}(\dot{\theta}_d - \dot{\theta}) \tag{1.1}$$

to a simulated 1m pendulum, with a desired sinusoidal trajectory of amplitude 0.2 radians and period 1s. This control strategy amounts to a critically damped proportional-derivative control policy with proportional gain k (see Section 2.5.5). Each time step, a simple simulation was updated using Euler integration, which is similar to many physics engines, although more sophisticated integration schemes such as Runge-Kutta are used in some (Erez et al. 2015). The root mean square tracking error in θ over 5000 time steps was measured for a range of gain values, for 60 and 2000 time steps per second, as shown in Figure 1.1.1. A rate of 2000 time steps per second or higher is often used in existing work (for example Yin et al. (2007), Coros et al. (2010), Wang et al. (2009), Liu et al. (2010), Wu & Zordan (2010), Wu & Popović (2010) and many others discussed in Chapter 2), while 60 time steps per second is more consistent with typical game simulations. For a given gain, the smaller time step leads to closer tracking, and the smaller time step also allows much larger gain values before control becomes unstable. Even for this simple example, the smaller time step makes control significantly easier. By comparison, a real locomotion simulation is 3D, has a more complex target trajectory, including contact events and discontinuous behaviour, and lacks the inherent stability of the pendulum. Furthermore, at 60 time steps per second, the duration of some important locomotion phases such as the period between the heel landing and a firm stance being reached will only cover a few steps of simulation. Therefore, it is reasonable to expect that locomotion control will be significantly more difficult with large time steps.

Chapter 3 illustrates the difficulties of large time steps in a more realistic problem, the implementation of a handcrafted controller based on SIMBICON, a well-known and often cited strategy from existing work.

Control strategies for simulated characters often use reference motions from motion capture or other sources as the basis of their desired behaviour. However, due to artefacts in the process of generating motions and retargeting them to the simulated character, such motions may contain physically implausible behaviour, such as ground penetration, floating or poor balance. Without addressing these physically

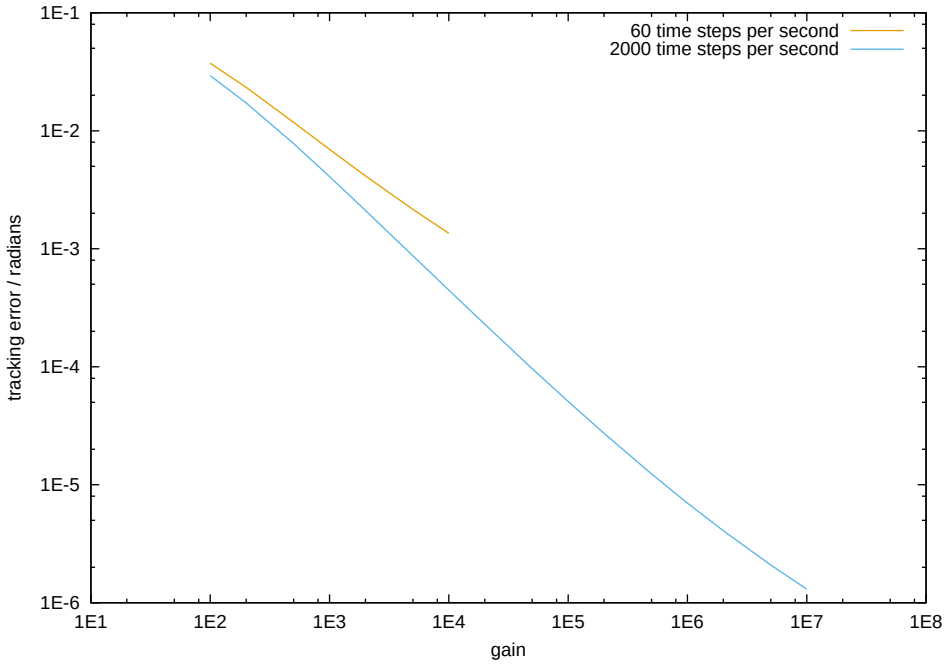

Figure 1.1.1: Effect of Time Step on Pendulum Tracking

inconsistent elements, physically simulated control based on such motion often fails. Chapter 4 describes the use of trajectory optimization to modify reference motions to eliminate such physical inconsistencies. The major contribution here is applying a previously unused approach to solve contact constraints, allowing contacts to be varied during optimization for 3D problems.

Once a physically valid reference has been generated, simulated control strategies can be formulated. An open loop control strategy which tracks the reference motion in the absence of disturbances is beneficial as it means that a separate feedback controller can concentrate solely on correcting deviations from the ideal trajectory, without having to also generate the desired base motion. Chapter 5 describes an existing sampling-based approach for generating open loop control, and the modifications required to operate with large time steps and with more typical game physics simulation software. Additions here include introducing a strategy to optimize the dimensions of the sampling space, successfully demonstrating that the technique may be modified to use much larger time steps than previous work, and introducing a method to generate cyclic control policies.

As described above, an open loop control policy is only useful in the absence of disturbances, and a feedback controller is required to deal with deviations from the ideal trajectory. The adaptations required to use a feedback controller based on evolutionary optimization with large time steps are described in Chapter 6. For interactive applications, it is also useful to be able to detect when a disturbed character state will lead to controller failure, and Chapter 6 describes a machine learning approach to predicting controller success based on the character's current

state. Contributions of this chapter include demonstrating that a successful feedback controller can be obtained with sufficiently low CPU usage (4%) to be applicable in games, and showing that a previously unused variant of a machine learning technique can be applied to make accurate (up to 98.9%) predictions of controller success with very low computational cost (order of microseconds).

1.2 Background on NaturalMotion

NaturalMotion was formed as a spin-off in 2001, building on research at Oxford University into biped control using neural networks and evolutionary methods. The company now employs around 300 staff, and has offices in Oxford, London and San Francisco. The initial focus of the company was on producing sophisticated animation tools for the video game and film industries. The main products on the technology development side are morpheme, an animation middleware tool, and euphoria, which allows real-time simulation of biological behaviours using techniques marketed as "dynamic motion synthesis". The company also previously produced endorphin, a tool which uses a similar approach to euphoria, but in an offline fashion, to generate virtual stunts for film. As well as developing animation technology, the company began producing games in 2007, releasing its first title in 2009 and forming a separate NaturalMotion Games division in 2010. The games division now focuses on developing free-to-play mobile games with high production values.

morpheme is an animation system comprising a runtime engine and a tool for visually authoring animation networks. The visualization tool provides an intuitive, node-based, drag and drop interface to control how motions are combined at runtime using structures such as blend trees and state machines. During authoring, the output of animation networks can be examined exactly as it will appear in the runtime, and powerful debugging tools are provided. The software also facilitates integration of physics and rigid body simulation. Further features include advanced inverse kinematics routines for tasks such as pointing, grasping, and modifying foot positions for uneven terrain, and retargeting of animations at runtime to support multiple characters. PC, console and mobile platforms are supported. morpheme has featured in a number of high profile games, such as Enslaved: Odyssey to the West, EVE Online, BioShock Infinite, DmC: Devil May Cry, Dark Souls II, The Evil Within and Until Dawn. There are competing animation engines which offer some similar features, such as Havok Behaviour, and many game engines such as Unity provide graphical tools for controlling blends and transitions, but morpheme is widely regarded as the most mature and sophisticated product of its kind.

euphoria is a system for dynamically generating motion which responds to a char-

acter's environment, creating unique output and richer interaction. The system comprises a collection of units known as behaviours, which are adaptive, high level control modules to perform actions such as balancing, staggering or protecting the character. Each module collects information about the character's environment and uses AI routines or other approaches to determine appropriate actions, which are executed using physics simulation. Individual modules can be combined to construct complex character behaviour. The parameters and influence of each module can be controlled, and modules provide feedback to inform game logic. As the control provided in euphoria is complex, it has largely been limited to PC and more powerful console platforms. The technology has featured in games such as Star Wars: The Force Unleashed and its sequel, and a series of titles from Rockstar Games including Grand Theft Auto IV, Grand Theft Auto V, Red Dead Redemption and Max Payne 3.

NaturalMotion began developing their own games in 2007, and released American football titles under the name Backbreaker in 2009 on mobile and 2010 on console. The games take advantage of the company's animation technology and physics integration, and place emphasis on delivering impactful and unique tackles. The mobile version was more successful than the console game. A separate NaturalMotion Games division was formed in 2010, and now focuses on mobile platforms, in particular iOS. The company has also moved to a free-to-play business model, in which base versions of games are free, but additional features and accelerated game progress may be purchased, and aims to distinguish its games in the free-to-play market with high production quality. Several very successful titles have been released, including My Horse, CSR Racing and Clumsy Ninja. NaturalMotion was acquired by Zynga in February 2014.

Chapter 2

Literature Review

2.1 Introduction

Physics-based character control is of interest to a number of disciplines, but with different goals in each case. Biomechanics may focus very highly on physical realism. In robotics, real-time control, robustness and stability are important, while motion naturalness may be a secondary goal. For interactive, real-time applications such as games, visual plausibility rather than perfect realism is required, but fast and consistent performance is needed to achieve the desired frame rate, and responsiveness to uncertain user input is required. Movie effects require a higher degree of fidelity, and while performance demands are not as severe as for games, time is still important due to production constraints. This review focuses on approaches suitable for games, but also discusses work from other fields.

At present, character control in real-time applications is typically accomplished largely using kinematic methods. Although there has been much progress with kinematic approaches, they have several limitations. The number of animations required to span a motion space increases exponentially with the number of parameters describing the space, placing great demands on animators or motion capture. Some modifications to animations are possible at runtime, such as using inverse kinematics (IK) to control footstep locations, but actions which are too far from the set of predefined animations will be impossible to represent. Scripting responses for interactions with other objects and characters is also possible, but again entails much work for animators, and since every configuration cannot be predicted in advance, interaction animations tend to be very repetitive.

Physics-based character methods have the potential to overcome many of these problems by generating suitable motion for a given scenario based on control principles,

and producing lifelike and varied responses to interactions which automatically reflect the subtle differences in input conditions. However, there are many difficult problems associated with providing physical character control. Typical character models have dozens of degrees of freedom (DOF), with the result that the space of possible control strategies is very high dimensional. Furthermore, events such as footsteps create discontinuities in this search space, making selection of good controllers very difficult. Characters are underactuated, in that they have no direct control over the acceleration of their centre of mass (COM), but must achieve control indirectly through manipulation of ground reaction forces (GRF). Bipedal characters are also unstable, requiring constant active control to avoid falling. Finally, character control may have many potentially competing objectives, including naturalness, robustness to unexpected events, and agile response to user inputs.

Physics-based character control draws on research from many areas including animation, biomechanics and robotic control. This review describes the approaches which have been considered, and their successes and limitations. The remainder of this document is structured as follows. Section 2.2 gives a very brief overview of physics simulation, character representations and basic balance control. Section 2.3 discusses some aspects of naturalness and perception which may inform controller design. Methods to reduce the dimensionality of control problems are discussed in Section 2.4. A review of existing work is given in Section 2.5. Recent trends and open questions are discussed in Section 2.6.

2.2 Background

This section gives a very brief overview of physics simulation and physics-based representations of characters. For a much more complete discussion, see the review by Bender et al. (2014). Basic concepts in balance are also covered.

A physics engine determines the future state of a system based on its current state and the forces and torques applied and in a fashion consistent with the laws of physics, a process desribed as forward dynamics. Once the forces and torques have been determined, the future state is evaluated by numerically integrating the equations of motion (EOM) of the system in small time steps. A typical simulation loop is illustrated in Figure 2.2.1. It is also possible to calculate the forces and torques required to achieve a given motion, a process known as inverse dynamics, although this is typically more costly than forward simulation.

At each time step, the physics engine determines whether bodies have collided with each other and applies appropriate measures to prevent interpenetration. This as-

pect of simulation is particularly important for control, as a character's centre of mass motion can only be controlled via ground reaction forces. Contact handling may be achieved using penalty forces which act in a direction normal to the collision surface to push the colliding objects apart, or by constraint based methods. For penalty methods, tuning stiffness values to avoid penetration or bouncing is difficult, and high stiffness values may introduce instabilities with large time steps. Constraint based methods require iterative solvers, but efficient methods capable of handling hundreds of contacts are available. The proportion of kinetic energy retained in a collision is described by a coefficient of restitution. See for example the thesis by Mirtich (1996) for a more detailed discussion of collision detection methods and responses.

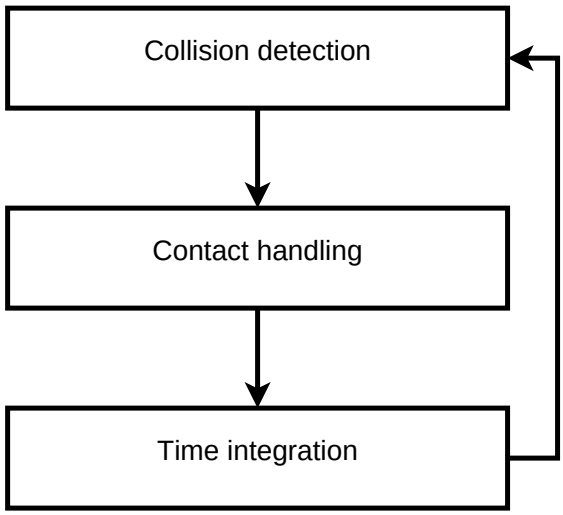

Figure 2.2.1: Typical Physics Simulation Loop

Friction is typically incorporated into physics engines using the Coulomb model. In this model, friction may exert a force in the plane of two touching bodies of magnitude up to $\mu f_\perp$, where $f_\perp$ is the normal force between them and μ is the coefficient of friction, a quantity which describes the roughness of the contact between the two surfaces. If the tangential force $f_\parallel$ is greater than the maximum frictional force, $f_\parallel > \mu f_\perp$, then sliding occurs. Because the frictional force is symmetrical about the normal direction, the set of forces which will not cause sliding is described by a cone, although this is often approximated as a pyramid in real-time applications for performance reasons.

Characters are generally represented in physics simulations as a hierarchical collection of rigid bodies. For each link, the resistance to linear and angular accelerations is described by an inertia tensor. Links are usually treated as having uniform mass distribution, with density values determined from measurements on humans. The motion of the character may be determined using maximal co-ordinate methods (Baraff 1996), in which joint constraints are explicitly enforced, with a redundant variable for each constraint. In this approach, care is required to avoid constraint

drift. Alternatively, reduced co-ordinate methods (Baraff 1996) may be used, in which variables are formulated to implicitly enforce constraints, and there are only as many variables as DOF. Efficient reduced co-ordinate methods such as Featherstone's algorithm (Featherstone 2014) are available, but typically involve convoluted algebra.

Connections between links may be described by various types of joint. Common joint types include hinge joints, which have a single DOF and might be used to represent joints such as knees and elbows, and ball-and-socket joints, which have three DOFs and are often used to represent joints with a greater range of motion such as the shoulder. Joint limits are often represented by separating the motion into a twist component about the long axis of the link, and a swing component capturing the other two DOFs, with motions constrained to lie within a wedge of twist values and an elliptical cone of swing values. Actuation of DOFs is typically treated as if there were a servo associated with each joint. These representations include many approximations necessary for real-time performance. Much more advanced models exist in biomechanics research, including actuation by pairs of muscles working antagonistically, more sophisticated joint limits, correlations between joints and modelling of elastic properties of tendons and other structures.

Many different physics simulation software packages are available and some of the most common options are discussed here. Open Dynamics Engine (ODE) is an open source engine which is commonly used in research (Geijtenbeek et al. 2011). It includes the option of a slow but very accurate solver, or a faster less accurate one. Bullet is another open source engine (Geijtenbeek et al. 2011). PhysX and Havok are two engines commonly used in games, where visually plausible results are typically sufficient and more severe approximations may be made to improve performance and robustness, such as artificial damping to prevent instability and ignoring Coriolis forces (Erez et al. 2015). Boeing & Bräunl (2007) give a good comparison of available physics engines and describe some shortcomings including integration errors, inaccuracies in friction calculations and poor simulation of collisions with low coefficients of restitution. The strengths and weaknesses of various physics engines are also discussed in Erez et al. (2015). NaturalMotion primarily uses PhysX for simulation, and that engine was used here for compatibility.

One of the most basic aims of control is maintaining balance. Common concepts in balance are the support polygon, the centre of pressure (COP), and the zero moment point (ZMP). The support polygon or support region describes the convex hull of the parts of the character in contact with the ground. The centre of pressure is the point through which ground reaction forces (GRF) may be considered to act. As described in Vukobratović & Borovac (2004), if a point exists within the support polygon at which the horizontal components of the net moment of all the

forces acting on the character is zero, then that point is known as the ZMP and, assuming there is sufficient friction to prevent sliding, the character is dynamically balanced. If the ZMP does exist (i.e. is inside the support polgon), then it coincides with the COP. If the point satisfying the moment condition is outside the support polygon, it is not meaningful as a true ZMP, as the reaction force must act within the support, but it may still be useful to consider it as a fictitious ZMP. Many robot control strategies depend on ZMP manipulation (Vukobratović & Borovac 2004), and these approaches have been shown to be effective for balancing and walking, but handling of disturbances is limited, and control tends to appear stiff and unnatural. These approaches also demand very precise, high frequency actuation, with very high energy requirements (Collins et al. 2005).

2.3 Realism

2.3.1 Characteristics of Natural Motion

This section describes some features observed in biological motion which may help to improve realism in simulated control strategies.

Formation of arm trajectories for point to point motions was studied by Abend et al. (1982). They found that hand trajectories when reaching for targets tend to be fairly straight and have a simple bell shaped velocity profile, whilst individual joint velocity profiles are often more complicated, which suggests that biological control is highly co-ordinated, and may be formulated in terms of end effectors.

Some characteristic invariants of human motion are described by Gibet et al. (2004). For many aiming type motions, there is a logarithmic relationship between speed and accuracy known as Fitt's law (Gibet et al. 2004), although the error may be constant or linear in some cases that do not use feedback based corrections, or where velocity is constrained (Bodenheimer et al. 1999, Bongers et al. 2009). Another characteristic invariant is described by the two-thirds power law, a power law relationship between angular velocity and curvature for hand trajectories, with the exponent 2/3. One potential explanation for this is given by Gibet et al. (2004) as minimization of jerk (rate of change of acceleration), but the biological significance of this rationale is not obvious (Harris & Wolpert 1998).

Harris & Wolpert (1998) explain the observation of bell-shaped velocity profiles and the above invariants and achieve convincing agreement with measured control signals for human hand and eye motions by assuming that the motor system exhibits noise proportional to the strength of the control signal, and that control is performed in

such a way as to minimize the error in position. Motor noise is also one source of variability in real motion (Wang et al. 2010), which is important as repetition is quickly noticed as unnatural (Bodenheimer et al. 1999). Noise at individual joints is not independent, for instance in experienced marksmen the shoulder and wrist joints act co-operatively to reduce aiming error (Arutyunyan et al. 1968, 1969).

Human locomotion typically involves little angular momentum about the centre of mass, with the arms acting to cancel the contribution from the torso and legs for instance (Herr & Popovic 2008). During locomotion, the head tends to have a smooth trajectory with only small lateral and vertical motions, for reasons thought to be related to stabilizing the visual and balance systems (Pozzo et al. 1990, Wampler & Popović 2009). There is a tendency to favour particular joints over others, depending on the task. For example, Novacheck (1998) gives work ratios for the hip, knee and ankle joints in walking, running and sprinting, and in particular finds that the knee is largely passive in walking.

Another characteristic of biological control is that low stiffness is typically exhibited for familiar motions (Takahashi et al. 2001). Stiffness may be varied in a task dependent manner, for instance when running on different surfaces (Farley & Morgenroth 1999).

Control systems in humans have latencies which are significant compared to the timescale of moderately fast motions, around 30-50ms for spinal feedback or 150-250ms for visual feedback (Kawato 1999). Such large delays would hamper feedback control, and suggest that biological control may include a significant feed-forward component. Artificial stimulation of the vestibular system has significantly greater effect on footstep placement when applied during the stance phase than at other points in the locomotion cycle, suggesting that footstep locations are largely planned in advance before steps are initiated (Bent et al. 2004). There is strong evidence that biological systems use learned simulations of their dynamics, called internal models, to accomplish feed-forward control (Kawato 1999, Takahashi et al. 2001). There is also evidence that internal models adapt to environmental influences. For example, hand motions are initially disturbed when external force fields are applied, with the disturbance diminishing over time, and equal and opposite displacements are initially observed when the external force is removed, suggesting that trajectories are planned in advance using internal models which are adaptive (Shadmehr & Mussa-Ivaldi 1994).

Biological control is regulated by the nervous system, which comprises the central nervous system (CNS), including the brain and spinal cord, which orchestrates high level control, and the peripheral nervous system which connects sensory receptors, organs and muscles to the CNS. The nervous system consists of a large number of

neurons, individual units which are densely interconnected. The activation of each individual neuron is determined by the collective influence of its inputs from other neurons, which may act to excite or inhibit its activity.

In many animals, rhythmic activities such as breathing, heartbeat and locomotion are thought to be governed to a significant extent by central pattern generators (CPG), neural circuits which generate oscillatory outputs without requiring varying inputs, sensory feedback, or high level control (Matsuoka 1985). There is also evidence for CPGs in humans, although supraspinal control and sensory feedback appear to be more important than in other animals (Luksch et al. 2010). Ivanenko et al. (2004) present measured muscle activation patterns for several different subjects at a large range of walking speeds (1-5 km/h) and find that about 90% of the activation can be explained in terms of 5 basic feed-forward patterns, which is consistent with the idea that CPGs generate much of the activation. It is also thought that these basic components may be reused for many different motions, for instance it appears that walking and running are composed from the same patterns, only differing in timing and intensity (Cappellini et al. 2006). CPGs are often modelled using artificial neural networks with mutually inhibiting neurons (Matsuoka 1985). Matsuoka (1985) also suggests that adaptation, in which the output of a neuron in response to a step input increases initially and then decreases over time, is significant in producing oscillatory behaviour. He presents a model of neuron activity including this feature, and gives conditions on the model parameters for oscillatory outputs to be produced.

Reflexes allow feedback control through responses to sensory input. In common usage, the term reflex refers to an involuntary response to a stimulus, but the technical definition also includes more complex, modulated responses, which may be dependent on task, motion phase or stimulus intensity (Zehr & Stein 1999). Monosynaptic reflexes involve only one sensory neuron and one motor neuron and are very fast, with time delays of order 50ms. The stretch reflex, which causes a contraction when a muscle is rapidly stretched, is an example of a monosynaptic reflex (Zehr & Stein 1999). Geyer and colleagues suggest that a reflex generating positive force feedback is important in leg compliance (Geyer et al. 2003, Geyer & Herr 2010).

There is evidence that various aspects of control in human walking are dependent on the locomotion phase. For example, the modulation of reflexes is phase dependent (Zehr et al. 1998). Luksch et al. (2010) suggest that the compliance of certain joints may vary depending on locomotion phase, for example the ankle stiffness may be lowered prior to heel strike to allow adaptation to uneven terrain, with higher stiffness during subsequent phases to ensure a robust stance. Bent et al. (2004) examine the effect on foot placement of galvanic stimulation of the vestibular system, and find that stimulation has a significantly larger effect during the stance

phase than during other phases. Similar effects are observed with visual (Hollands & Marple-Horvat 1996) and somatosensory (Zehr & Stein 1999) input. Luksch et al. (2010) suggest that the 5 activation patterns observed by Ivanenko et al. (2004) may be associated with 5 phases of the locomotion cycle (weight acceptance, propulsion, stabilization, leg swing, and heel strike).

There is evidence that the mechanical properties of muscles may play an important part in biological control. The Hill muscle model, illustated in Figure 2.3.1, is commonly used to represent these properties. The model consists of a contractile element, which generates the active force, a series element representing the intrinsic elasticity of the muscle and tendons, and a parallel element representing the effect of connective tissues. The model also includes force-velocity and force-length relationships which reflect the properties that muscles can apply force more effectively near their optimum length, and applied force diminishes with velocity. Zajac et al. (1989) give a review of the Hill and other muscle models.

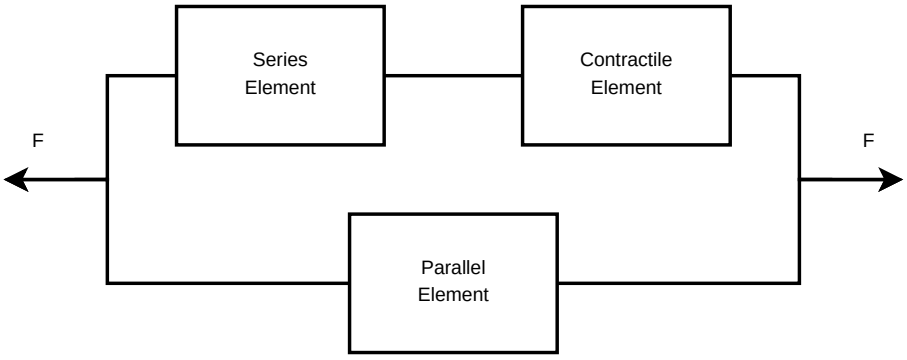

Figure 2.3.1: Hill Muscle Model

van Soest & Bobbert (1993) look at explosive movements such as throwing, kicking and jumping, where the motion time is so short in comparison to neural latency that control signals must be largely preprogrammed. They consider two open loop control policies, one in which joint moments are directly prescribed, and one which specifies muscle activations, with joint moments arising as a function of muscle properties. They find that control via directly specified joint moments is significantly more sensitive to disturbances, and suggest that the force-length-velocity relationship of muscles can be considered as an auxiliary feedback control system with zero latency. Similarly, Wagner & Blickhan (1999) find that muscle properties can stabilize vertical oscillations of the centre of mass in a simple model of human leg-bending, without changes in muscle activation. Pratt & Krupp (2004) suggest that adding elastic elements between drive and load in legged robots may introduce desirable characteristics such as shock tolerance, low impedance, and good force control over a wide range of control frequencies.

The equilibrium point hypothesis (Bizzi et al. 1992) suggests that biological control

is achieved by manipulating the stiffnesses and force-length relationships of agonist-antagonist muscle pairs to alter the rest position of joints. Control is then formulated as a sequence of equilibrium positions and joint stiffnesses, with the spring-like properties of the muscle providing stabilization.

A common hypothesis in research is that biological control may be formulated in terms of optimality principles. Alexander (2001) notes that evolution and learning are two strong mechanisms in nature likely to lead to optimality. Many successful applications of optimality principles in explaining biological behaviour are described by Todorov (2004). Various natural bipedal gaits can be discovered by optimization methods without prior information about motion (Wang et al. 2012, Mordatch et al. 2013, Geijtenbeek et al. 2013). Anderson & Pandy (2001) obtain impressive agreement with measured trajectories and muscle activation patterns for a walking motion by optimizing energy usage per unit distance using a detailed model of a human leg with elaborately simulated muscles, although the optimization is extremely expensive computationally.

There are various potential drawbacks to employing optimality principles. The quantities being optimized in nature are unknown and potentially extremely complicated, and it is debated how useful a simplified objective function can be. There are many competing theories of what variables are optimized in natural systems, for example minimization of energy consumption, change in joint torque, and acceleration have all been suggested (Todorov 2004). It may be very difficult to define objective functions which capture motions with stylistic elements, such as a sad or happy walk. Alexander (2001) also notes that an important difference between natural evolution and controller optimization is that each predecessor of a real creature must have been viable, which may constrain natural evolution to local optima. Although optimization produces realistic results using the sophisticated simulation used by Anderson & Pandy (2001), real-time applications require much more basic models of actuation, joint limits, and elastic energy storage mechanisms and it is not clear how well optima for simpler representations will correspond to real systems (Liu et al. 2005). Optimization is easier to apply to high energy motions such as running, jumping and diving, which are highly restricted by dynamic constraints. Low energy motions such as walking may have many physically correct solutions and are poorly constrained by optimization alone (Liu et al. 2005).

2.3.2 Perception

This section describes some perceptual factors which may strongly influence the acceptance of a character animation as natural or otherwise.

Motion is very important in cognition. For example, very sparse point light representations are instantly recognizable as human when dynamic, but not when statically displayed (Johansson 1976). Friends can be recognized based solely on gait, using point light displays which suppress other potential identification cues (Cutting & Kozlowski 1977). People are very adept at recognising and interpreting motion of humans and other animals for evolutionary reasons, as this skill is important for numerous reasons including social interaction and survival (Johansson 1973, Troje 2002). Furthermore, people will have encountered a huge number of positive examples of natural behaviour (Ren et al. 2005). Perception is affected by the level of anthropomorphism of simulations. For example, classification algorithms can perform better than humans at identifying gender from gait for a point light representation (Troje 2002), people are more sensitive to time distortions in animations of detailed geometrical models than stick figures (Pražák et al. 2010), and subjects are more sensitive to small differences between pairs of animations for polygonal models than stick figure models (Hodgins et al. 1998). This is important to bear in mind when evaluating existing work on character controllers, which may vary significantly in rendering style. Representations of humans which are close to being lifelike can fall into the "uncanny valley", in which small defects from realism can be very noticeable and offputting, particularly differences in motion (Mori 1970, Hodgins et al. 2010). Hence, problems with animation can destroy the immersiveness of otherwise engaging simulations.

There is some evidence that realism may influence perception at a subconscious level. For instance, Oesker et al. (2000) study perceived skill of robotic football players animated at various levels of sophistication. Participants in the study do not report explicit awareness of difference in the animations, but do tend to attribute a greater level of skill to more elaborate animations. Thus, users of applications may be aware that something is not right with a character's motion and it may affect their level of acceptance even if they cannot put their finger on the exact problem.

For many applications, perfect accuracy is not required and plausibility is sufficient (O'Sullivan et al. 2003). Generally in real-time applications there is a trade-off between simulation accuracy and performance. Many researchers have investigated perceptual thresholds for various simulation errors with the idea of establishing quantitatively what trade-offs are acceptable. Sensitivity to individual errors generally decreases with scene complexity and dimensionality (Gilden & Proffitt 1989, O'Sullivan & Dingliana 2001). Reitsma & Pollard (2003) study sensitivity to errors in ballistic motion of animated humans and find that errors in horizontal motion are more likely to be detected than those in vertical motion, and that sensitivity is greater for increases in acceleration than decreases. Research on perception of motion time warping by Pražák et al. (2010) suggests that users are more likely

to detect walking animations that have been sped up than those that have been slowed down. Ren et al. (2005) also note that humans may be relatively insensitive to slowed down motion, unless clues such as reduced gravity in flight phases are present.

Much existing work regards realism as an important objective, and attempts to preserve it by restricting edited or generated motions to be in some sense close to reference motion, but the achievement of this objective is often judged subjectively, and there is relatively little work on systematically evaluating naturalness. One example of a quantitative approach is by Ren et al. (2005), who consider a variety of statistical measures to classify motion samples as realistic or unnatural. A method based on an ensemble of hidden Markov models is found to perform relatively well, although still significantly worse than classification by humans. The ensemble approach also has the advantage that it uses a hierarchical decomposition of the character model which is able to pinpoint the location of unnatural motion. It is suggested that such a classification scheme could be used to constrain the output of motion controllers.

2.3.3 Techniques to Improve Realism

This section describes some approaches to increase the realism of animations.

Physical accuracy may be improved by post processing with filters that ensure physical validity, for example to enforce valid ground contacts (Pollard & Reitsma 2001) or conservation of angular momentum and correct COM trajectories for flight phases (Shin et al. 2003). Safonova & Hodgins (2005) describe some considerations to improve the physical validity of blended animations, including interpolating the COM rather than the character root, and using a weighting scheme designed to preserve correct gravity forces.

Physical correctness alone is not sufficient to ensure natural appearance. A common approach to improve naturalness is to track a reference motion capture sequence. Motion capture has the advantage that the raw motion is guaranteed to be physically feasible, although the physical validity may be lost in processing unless care is taken. It is also good for capturing lifelike motion with the desired style. Since there are almost certainly discrepancies in morphology between the motion capture actor and the animated character, the motion must be retargeted, which can increase the difficulty of tracking. Differences in leg lengths between capture actors and simulated characters can cause tripping (Sok et al. 2007). Differences between the simplified actuation and friction models used in simulation and the real capture environment may also cause problems. Naïve tracking of motion capture data will

also often result in instability due to accumulation of errors. One further problem with motion capture data is that GRF are typically not measured, so one source of important control information is missing.

Noise may be added to animations to increase variability (Perlin 1995). However, due to the co-ordination of noise between joints, care must be taken to avoid unnatural results (Egges et al. 2004). One method to deal with this difficulty is to add noise to an orthogonalized basis, such as that obtained by principle component analysis (PCA). Physical simulation has the advantage that some noise is inherently present, due to finite precision, approximations, varying time steps and so on.

Some methods incorporate delays in feedback to simulate biological latency (Kwon & Hodgins 2010, Jain et al. 2009, Wang et al. 2012). Feed-forward control may also be used (Yin et al. 2007), although it may not cope well with unexpected disturbances.

There is often an intuitive expectation that techniques such as genetic algorithms and neural networks which mimic natural processes may be inherently more likely to produce lifelike results, but there is little rigorous justification for this idea.

2.4 Dimension Reduction

As noted earlier, typical human character models have dozens of DOFs, and this high dimensionality contributes to the difficulty of control. This section describes some approaches which ameliorate this problem by considering systems with fewer DOFs.

Many control systems use simple variables which summarize the system dynamics, such as the COM linear momentum and angular momentum (Macchietto et al. 2009) or the ZMP (Kajita et al. 2003). Popović & Witkin (1999) use manually created low DOF models to reduce complexity. Their approach is justified by biomechanical insights from Blickhan & Full (1993), who examine locomotion for a variety of creatures and find strong similarities in energetics, gait and ground reaction force despite great diversity in morphology, and suggest that a single legged mass on a spring model can capture many of the important characteristics of locomotion. Dimension reduction techniques employed by Popović & Witkin (1999) include removal of insignificant DOF, such as elbows and wrists in running and walking, replacement of whole subtrees of the character model with point masses, and exploiting symmetry. da Silva et al. (2008*a*) also use a manually created low DOF model consisting of two leg links and a torso link, to suppress less important DOFs when determining control policies. Similarly, Wu & Popović (2010) formulate control in terms of a reduced

DOF model, with the most important information about the dynamics captured by positions of end effectors for the two feet and the upper body.

Coros et al. (2008) use a manually defined reduced model based on the COM position and speed, and the torso and hip angles. These variables are chosen as they are associated with the heaviest links and high level characteristics of motion, allowing the dimensionality to be lowered while retaining the essence of the state description.

Safonova et al. (2004) argue that since human motion is highly co-ordinated, there are many correlations between DOF and high dimensionality is merely a result of an inefficient choice of representation. They find that a reduced basis of 5-10 DOFs which still retains most of the important information can be determined for a particular type of motion by performing PCA on a set of similar motions. An IK post-processing step is required to ensure constraints such as footplants are met, as it may not be possible to satisfy them exactly in the reduced basis. Carvalho et al. (2007) compare reduced representations determined by PCA and probabilistic PCA for golf swing motions and find that PPCA can require less computation to satisfy constraints, but it tends to produce more motion artefacts than PCA. Reduced bases determined in this manner may not be very intuitive and may be strongly dependent on example data. Safonova et al. (2004) note a failure using a PCA approach with only one example motion, presumably because there is insufficient variety in a single motion for the reduced basis to adequately span the motion space.

The inverted pendulum (IP) model is a low dimensional representation that is often used to describe the motion of the stance leg during the single stance phase of walking (Coros et al. 2010, Tsai et al. 2009). The COM is treated as a point mass connected to a light, inextensible pendulum with its origin at the COP. The inverted pendulum model is used to describe the COM trajectory, and calculate target positions for the swing leg. Various extensions to the IP model exist. In the spring loaded IP model (SLIP), the restriction of constant leg length in the IP is relaxed by treating the stance leg as a spring, allowing elastic energy storage and release mechanisms to be modelled (Mordatch et al. 2010). The SLIP model has the disadvantage that its equations of motion are not analytically integrable, making it costlier to evaluate. Modelling of angular momentum is incorporated in the angular momentum inducing IP model (Komura et al. 2004) and IP plus flywheel model (Pratt et al. 2006). Other related models are the inverted pendulum on a cart (IPC), in which the inverted pendulum is connected to a cart by an unactuated joint and control is accomplished by forces applied to the cart (Kwon & Hodgins 2010), and the similar cart-table model (Kajita et al. 2003). Kwon & Hodgins (2010) suggest that the IPC model may be more suitable for running than the IP model, since the origin of the pendulum in IP is poorly defined during flight phases, and IPC avoids discontinuities.

For running, absorbing and releasing energy is important, and a mass-spring model is often used. Geyer et al. (2006) argue that compliance is also important in walking, as the stiff legs of the inverted pendulum model produce exaggerated vertical motion, and cannot reproduce the characteristic doubly peaked GRF observed in human walking. They propose a model based on a mass with a spring corresponding to each leg, which more accurately reproduces these motion features while retaining relative simplicity. Using this representation, walking and running can be considered as two modes of a unified model.

In robotics, simple passive dynamic walking systems, which can achieve downhill locomotion without actuation, are often used to gain insight into dynamics (McGeer 1990, Tedrake 2004). Examples include the rimless wheel model and the compass gait model. Passive dynamic walkers are also used to simplify control, as active locomotion can be achieved with the addition of a small number of actuators, and the underlying passive system has high inherent stability (Tedrake 2004). A similar principle may apply in biological control, for example Kubow et al. (2002) suggest that neural control in animals builds on their underlying passive dynamics, and that control effort is concentrated on dimensions which recover slowly or not at all via passive dynamics.

Low dimensional models are important in preview control planning for real-time applications (Mordatch et al. 2010, Kajita et al. 2003). Since it is helpful to plan over several steps, and plans must be recalculated frequently due to deviations, full DOF models would be computationally intractable in real-time.

2.5 Character Control Methods

2.5.1 Spacetime Optimization

This section discusses the idea of spacetime optimization (SO), in which motion is optimized over a whole trajectory.

The idea of manipulating motion by treating it as a constrained optimization problem (described variously using the terms trajectory optimization, spacetime optimization, spacetime constraints) is introduced in the work of Witkin & Kass (1988), with the idea that animators specify a minimal outline of desired motion, such as starting and ending poses and velocities, and desired characteristics for how the motion is to be performed, such as energy efficiency. A physically valid motion satisfying the pose constraints and meeting the other criteria as well as possible is then generated by solving a constrained optimization problem for the motion and

actuation over the whole motion duration. The aim is to reduce the amount of work required from the animator by automatically handling the mechanical process of ensuring physical realism, allowing the animator to concentrate on artistic aspects of the motion. The resulting constrained optimization problem is solved by a technique known as sequential quadratic programming (SQP). In SQP, iterative refinements to a solution are calculated using Newton iteration on a quadratic approximation to the objective function, and these refinements are projected onto the subspace of solutions which do not violate the constraints. Each of these steps boils down to a sparse linear problem, but since the motion and actuation are discretized over the whole interval, and their values at each time step are considered as independent variables, these linear problems may involve thousands of unknowns, precluding real-time performance. Witkin & Kass (1988) demonstrate co-ordinated jumps with a low dimensional model, which exhibit important characteristics of traditional animation such as anticipation and follow-through.

The search space for spacetime optimization is likely to be discontinuous with many local minima, so that methods such as SQP require initialization close to a desired trajectory to reach good solutions. Ngo & Marks (1993) address this problem using a genetic algorithm approach to consider a large number of trial solutions and evolve towards good solutions, at the expense of even greater computation time. Another approach is to provide several keyframes throughout the motion interval without specifying their exact timing, which also gives a much more constrained optimization problem with faster convergence (Liu & Cohen 1995).

Various work attempts to improve the performance of spacetime optimization. Cohen (1992) and many subsequent authors describe the motion using splines, greatly reducing the number of variables. Liu et al. (1994) use a wavelet based representation, allowing hierarchical refinement of motion detail only where necessary.

The complexity of calculating derivatives required by optimization methods such as SQP is generally quadratic in the number of DOF. Fang & Pollard (2003) improve performance by carefully formulating the problem to reduce the time required in evaluating derivatives. They show that if constraints and objectives are expressed in terms of aggregate forces and torques, and individual joint torques are not required, derivatives may be calculated in time which scales linearly with the number of DOF, rather than quadratically, reducing the necessary computation time at each iteration and improving scalability to complex characters. For a character with 22 DOF, a leap motion of duration 0.75s is optimized in 4 minutes. As individual joint torques are not explicitly calculated, the method excludes the use of certain forms of objective function, although alternatives can usually be constructed. However, efficient automated differentiation software is readily available now, so calculating derivatives is less likely to be a bottleneck.

Safonova et al. (2004) apply spacetime optimization to full human character models by using PCA on examples of similar motions to determine reduced bases of 5-10 DOFs. However, several example motions are required to provide sufficient generality in the low dimensional space, and each class of motions requires a separate basis. The approach is used to synthesize motions of duration around 2 seconds in less than 10 minutes, and to make substantial modifications to example walking, jumping, running and acrobatic motions, whilst maintaining realism.

Liu & Popović (2002) use a simplified model of character dynamics to reduce problem difficulty. Rather than a complete dynamics formulation, only the overall linear and angular momenta of the character are physically constrained, and an empirical model of the characteristic profile of momentum transfers is used rather than a complete model of contacts. With these simplifications, a number of highly dynamic motions such as running, jumping and gymnastics are synthesized from rough motion sketches for a character with 51 DOF, each taking less than 5 minutes to optimize. The simplified dynamics may not be as well suited to low energy motions such as walking. Abe et al. (2006) use a similar approach, but with a more general spline-based representation of the momentum, which is constrained to follow a pattern similar to an input reference motion. Optimization for a 43 DOF character takes 2-4 minutes.

Popović & Witkin (1999) use spacetime optimization to modify reference motions in a physically valid manner (unlike other motion editing methods), rather than generating motion from scratch. To cope with the difficulty in applying SO to high dimensional characters, low DOF models which retain the characteristics essential for particular motions are manually defined. The motion of the low DOF model is then modified using SO with an objective function chosen to minimize deviation from the reference, so as to retain the subtle details of the original motion. The modification to the low DOF motion is then mapped back to the full character. Significant changes to reference motions are demonstrated, such as crossing footsteps or limping in a running sequence, and jumping diagonally or over obstacles based on a plain jumping motion. As the example motions provide good initialization, performance can be relatively good (a few minutes) compared to other SO problems. This approach could be used to generate motion libraries from a small set of reference sequences.

Liu et al. (2005) analyse reference motion by assuming that it is the solution of a SO problem for some unknown set of parameters which encode joint stiffnesses and muscle preferences. The problem is then tackled backwards to estimate the parameter values most likely to have produced the reference motion. These parameters can then be used to generate new motions in a similar style to the reference with SO. Examples include modifying a walk to simulate carrying a weight, and generating

uphill or downhill motion in the style of a sad walk. The method also has the advantage of determining stiffness and muscle preference parameters, for which it can be difficult to find appropriate values manually.

These optimization problems are typically tackled using derivative-based solvers, and some features such as the timing of foot contacts are difficult to formulate in differentiable form. None of the work discussed above allows the details of the contacts to be modified during the optimization.

Liu et al. (2006) consider optimization of multi-character interactions, which would be impractical to manually schedule. The position and timing of interactions is modified during the optimization, with time warping used to change time of interactions, and constraints expressed in warped time. Both co-operative motions such as walking hand in hand and adversarial motions such as tackling are demonstrated. To achieve good performance with the very large problems resulting from multiple characters, variables are grouped into blocks which are solved separately while holding the remaining unknowns fixed. Subproblems for different blocks are then interleaved, and depending on the scheduling of subproblems the influence of each character can be varied. A continuation method is used to introduce interactions gradually, for example using a spring between the hands of two characters with spring strength increasing as the optimization proceeds. Complete solutions require up to 85 minutes, with each subproblem taking 8-20 minutes.

Wampler & Popović (2009) use spacetime optimization as an inner loop of a solver, with a stochastic, population-based method (covariance matrix adaptation (CMA), see Section 2.5.6 for more details) used to optimize timing of footfalls in an outer loop. The outer loop does not require derivatives, and as it uses a population of potential solutions it is better able to avoid local minima. Based on the interplay between motion and morphology, minor modifications to limb lengths and radii are allowed in the inner loop. Plausible gaits are generated for a variety of creature topologies (10-33 DOF), without prior information about their motion. The method requires hundreds of hours of CPU time, with each iteration of the inner loop taking 10-20 minutes.

To obtain a better behaved search space, Mordatch et al. (2012) handle contacts using soft constraints. Motion is split into a series of 10-20 0.5s phases, with contacts invariant during each phase. The activity of each potential contact during each phase is represented by a continuous variable, giving a smoother problem, with physically valid constraints encouraged via objective function terms. Because of the avoidance of discontinuities in the problem formulation, the optimization problem can be solved quickly using relatively simple methods, taking 2-10 minutes. Physics terms in the objective function are gradually introduced as the optimization pro-

ceeds, allowing flexible exploration of the search space at first. The soft constraints are not guaranteed to be satisfied, but are found to converge in all the examples they consider. Motions such as getting up, crawling and climbing are demonstrated, as well as co-operative actions such as passing objects between characters and one character climbing on another.

Todorov (2011) introduces a formulation for contacts which is invertible in the sense that the contact forces can be straightforwardly calculated from the character's rigid body positions and velocities on subsequent time steps. The contact formulation is continuous with contact force decaying with distance subject to a smoothing parameter. The smoothing parameters allows continuation methods, in which contacts can be softer in the early stages of solution. The invertible formulation allows trajectory optimization problems to be constructed with fewer variables and constraints. It also allows larger time steps to be used as potentially unstable forward integration can be avoided. The efficiency of the method is demonstrated by performing a single Newton step of trajectory optimization in 160ms.

Mordatch et al. (2013) use a similar method to Mordatch et al. (2012), applied to human lower body motions such as walking, running, jumping and kicking, using flexible control without task-specific structure. The explicit contact activation variables used in the previous work are removed, instead defining the activation of each contact implicitly as a step-like function of its normal force. This also removes the need to separate motion into contact phases. Biologically inspired actuation is included by adding a term to the objective function encouraging joint torques consistent with muscle physiology. The physical consistency and muscle consistency terms in the objective function are given large weights to obtain values close to zero in the solution. For a 36 DOF character, solutions require 6-10 minutes, and show good agreement with joint angle and moment values measured from human observations reported by Wang et al. (2012) (see Section 2.5.6). The muscle based objective gives more realistic results than torque actuation, which exhibits severe crouching.

Posa et al. (2014) show that contacts can be incorporated as variables in the optimization problem, allowing the solver greater flexibility and avoiding difficulties with prescribed contact schedules. Including contacts introduces complementarity constraints which increase the difficulty of finding a solution as the feasible region is poorly connected. The complementarity constraints are tackled using a constraint relaxation method. Only 2D examples are demonstrated, albeit with complex contacts. The most complex problems take up to an hour to solve for a planar model with 13 DOF.

In the work of Al Borno et al. (2013), trajectories are synthesized over a series of

0.5s spacetime windows using a shooting method, iteratively improving the control governing a forward simulation using CMA. Timings for contact are pre-defined, but precise positions are not required. A highly parallelized implementation requires about 20 minutes for each 0.5s window with a 41 DOF character.

Trajectory optimization has been used in the generation of motion graphs. Ren et al. (2010) use trajectory optimization to add transitions to a motion graph, which are ultimately accepted or rejected by a user to ensure the desired quality. A finite horizon search is used to detemine an optimal sequence of transition generations so as to minimize computation and demands on user evaluation, taking into account the improvement to the graph's connectivity and the estimated probability of the optimization succeeding. The probability that the optimization does not fail and is not rejected by the user is estimated based on the smoothness of COM and end effector trajectories in a simple interpolated transition. The approach results in a well connected motion graph with good responsiveness, which is able to handle more dissimilar transitions than other methods. Wampler et al. (2013) use a similar approach to synthesize motion graphs for creatures from scratch, based on a model of the range of tasks to be performed such as direction following or speed matching. A motion graph is constructed incrementally, beginning with a walk cycle generated using the method from previous work (Wampler & Popović 2009). Markov analysis of the task model determines the long run probability that each state in the motion graph is active, and motions are added to the graph based on an objective function averaged over the whole graph, weighted by probability. Plausible motion graphs are demonstrated for creatures with 16-29 DOF, requiring 8-10 hours for a 2 parameter task model.

Wampler et al. (2014) analyse a database of animal motions in order to predict new gaits for unknown creatures. Style parameters are extracted from each animal's motion and a regression model is used to interpolate to new creatures. Trajectory optimization is used for individual motions, and a CMA outer loop optimizes the regression parameters. The method gives plausible results for a wide range of creature sizes, and performs well at reproducing one of the input motions when it is excluded from the database. However, the approach is very expensive, requiring several days of computation on a 96 computer cluster.

There are numerous difficulties with SO approaches. Even with modern hardware, performance is nowhere near real-time, so they are only suitable for offline computation. It is very difficult to apply SO to high dimensional characters without good initialization, as the optimization methods tend to get stuck in local minima. As the approach optimizes a whole motion trajectory rather than a controller, it is not able to deal with unexpected disturbances, and is therefore unsuitable for interactive applications. It may however be useful for physically filtering motion sequences,

motion editing or motion synthesis.

2.5.2 Neural Networks and Biologically Inspired Control

This section describes approaches to developing control based on methods which mimic natural control processes.

Neural networks (NN), illustrated in Figure 2.5.1, and similar methods consist of a graph of nodes (or neurons) with directed connections between them. Nodes determine output values based on a weighted sum of their inputs, modulated by an activation function. The activation function is usually nonlinear to allow more complex behaviour, with the hyperbolic tangent or other sigmoid functions common choices. The final behaviour of the network is then determined by adjustments to the set of input weights. Selection of a suitable network topology is an important consideration in using neural networks, with more complex topologies allowing more complex behaviour but entailing greater computational expense and greater difficulty in determining appropriate weights. A common topology known as a multilayer perceptron consists of a layer of input nodes, which provide information about the system state or sensor data, one or more layers of hidden nodes to perform computation, and a layer of output nodes. Networks which include loops allow storage of an internal state, and are described as recurrent. Without recurrency the response of the network is a static function of its inputs (Allen & Faloutsos 2009). The topology may be fixed in advance, or altered along with the weights to achieve the desired output (Allen & Faloutsos 2009).

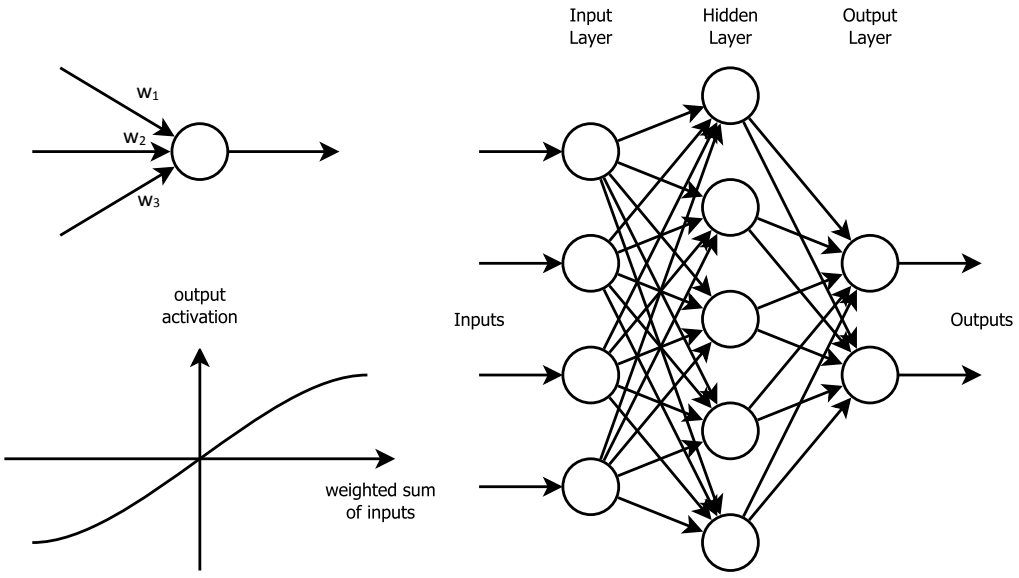

Figure 2.5.1: Example Neural Network Activation Function and Topology

In other applications, weights for neural networks are often determined by providing

a set of example inputs with ideal outputs. Weights which allow the network to most closely match the ideal outputs can then be calculated by gradient based methods. In the case of character control, sufficient example motions may not be available, or the goal may be to generate motion from scratch. In this case, it may be possible to find suitable weights by creating a large number of random networks, evaluating their performance using some fitness function, and refining the most effective ones, a so called generate and test approach. Since this approach requires only forward simulation, it can easily handle features such as contacts, friction and discontinuities which can be difficult to incorporate into other methods such as spacetime optimization (van de Panne & Fiume 1993).

A significant part of the work in the generate and test approach lies in designing a good fitness function. For locomotion tasks, the fitness function is often based on distance travelled in a given interval. Terms may be included to encourage straight line walking or keep the COM high to prevent crawling motions (Reil & Husbands 2002). It may also be useful to reward stability at the end of the interval, to eliminate controllers which might reach the end of the interval but fail soon afterwards. Complex goals are often achieved by first optimizing for a related simpler task and incrementally adding complexity using the previous solution as a starting point (Reil & Massey 2001). Controllers that are obviously unsuccessful are often terminated early to improve performance (Sims 1994, Reil & Husbands 2002, Allen & Faloutsos 2009). Capturing characteristics such as naturalness and style in fitness functions is extremely difficult (van de Panne 2000).

van de Panne & Fiume (1993) use sensor-actuator networks (SAN), similar to NN but with input based on sensor data rather than state, to generate diverse locomotion behaviours for low dimensional planar creatures. The nodes in the SAN are dynamic systems themselves, with time delays and hysteresis to prevent rapidly switching on and off. Controllers are generated using stochastic gradient ascent or simulated annealing. For the low dimensional creatures considered, 1-5% of random networks produce useful behaviour, but this fraction diminishes with creature complexity and the technique would be unsuitable for high dimensional characters. Promising solutions are fine tuned by adjusting parameters such as node switching delays, range of motion and actuator stiffnesses using stochastic methods.

Luksch et al. (2010) generate motion based on several ideas from biological control. Control is implemented using a collection of co-ordinated feed-forward torque patterns, analogous to CPG, and phase-dependent feedback units which function similarly to reflexes. Activity of individual units may be stimulated or suppressed based on events, sensory information, or the activity of other units. Setting stiffness and equilibrium points allows passive dynamics for some joints. Robust walking is demonstrated for a 3D simulated character with 21 DOF, including walking on

slopes and steps. The approach shows improved naturalness and energy efficiency compared to typical robot control.

Matsuoka (1985) shows that mutually suppressive neurons can produce oscillatory behaviour when they exhibit adaptation i.e. a reduction in response over time with constant stimulus. Dynamic systems can be stabilized by entrainment with Matsuoka oscillators. Liu et al. (2013) explore how a dynamic system governed by a Matsuoka oscillator can be adapted to different conditions by applying state space transformations which bring otherwise unstable states within the transformed basin of attraction. Transformations considered included offsets to state variables (for example to adapt to walking on different slopes), time scaling (for example to adjust stride frequency), and energy scaling. The approach can also be used to control transitions, by transforming the basin of attraction of the destination motion to include the end state of the source motion. Adaptation for a 2D walker to different walking speeds and steep slopes is demonstrated. Apart from lack of ankle actuation in the model, good correspondence is shown between the adaptation of the simulated walker's limit cycle and measured human motion. Simulation is performed using a variable time step integrator, which could be awkward for real-time applications due to its inconsistent computational cost per frame.

There are various advantages and disadvantages to control based on NN and similar models. The approach does not require prior information about motion, and can be used in a highly automated fashion. These methods are arguably closer to the way control is developed in natural systems, compared to other methods. Intrinsic variability in output may also contribute to perceived realism. Although a lot of computation may be involved in the optimization process, the output of an optimized network is typically efficient to compute.

However, there is no guarantee that the approach will generate good solutions, and the workings of an optimized network are likely to be inscrutable. The level of automation may or may not be advantageous, as control over the style of motions produced is limited, although one possible way to exert more control is to manually rate controllers rather than using a fitness function. Designing appropriate fitness functions and managing the trade-off between network flexibility and optimization complexity may be difficult problems. Changes to character morphology, environment or goal will frequently require further optimization, although initialization using previous solutions may be useful. Apart from following tasks (Sims 1994, Reil & Husbands 2002), there has been little work on interactive control of these systems. These techniques may be difficult to apply to high dimensional characters.

2.5.3 Evolutionary Methods

This section describes the use of evolutionary methods such as genetic algorithms which are often used to progress towards good solutions. Several papers described below use evolutionary methods to obtain good parameter values for neural networks or similar control models (see Section 2.5.2).

Genetic algorithms (GA) mimic natural evolutionary processes by maintaining populations of candidate solutions, and producing new improved solution generations by mutating or combining the best performing solutions of the current generation. Single runs of GA can lead to homogeneous results, so it can be helpful to perform many runs to allow sufficient variety (Sims 1994). Evolutionary methods do not require derivatives, and as they are population-based they are able to avoid local minima.

Sims (1994) uses an evolutionary approach to modify both a control network and the morphology of simple creatures to achieve walking, jumping, swimming and following tasks. Nodes in the control network can apply a variety of mathematical functions. Nodes can retain internal states, and are able to generate periodic outputs such as sine and saw waves even with static inputs. It is noted that these features probably result in a less accurate biological model than the sum-threshold nodes used in typical NN. The morphology is represented using a directed graph, and both the morphology and the control network are updated using a GA approach. Mutations allow new nodes to be added to the control and morphology graphs, and new connections between nodes to be established. Impressive results are demonstrated for low dimensional creatures, but major modifications of morphology are likely to be unsuitable for many applications.

Reil & Massey (2001) and Reil & Husbands (2002) consider biped walking control using a model of a central pattern generator (CPG), a mechanism thought to be involved in biological control for generating cyclic outputs for tasks such as locomotion, without higher neural control. The CPG is modelled using a NN with a high degree of recurrency, which promotes periodicity. The network is not aware of the system state, and control is achieved by generating target joint angles for a manually tuned proportional-derivative controller (see Section 2.5.5). The biped model has relatively few DOFs, in particular the upper body is represented by a single link. Weights and other parameters of the network are determined using a GA. From 100 runs of 120 generations, about 10% produce stable walkers, with some runs producing lifelike walking. Fluctuations inherent in the NN control improve realism by reducing repetitiveness. Following of a sound signal is also demonstrated by appropriate adjustments to the fitness function. No prior information about motion is

required.

Allen & Faloutsos (2009) aim to improve automation and expand the range of potential behaviour for a NN walking controller by allowing the network topology to evolve. This is achieved by allowing mutations to add new nodes or connections. To achieve sufficient complexity to generate walking motions the networks require hundreds of parameters. Controllers typically become unstable after 5-10m.

2.5.4 Optimal Control

This section describes a common framework known as optimal control, which is often used to formulate control for physically simulated characters.

In general, an optimal control problem involves selecting a control policy $u(t)$ over some time interval in order to influence a state $x(t)$ in some optimal fashion. In character simulation, the control could be joint torques, and the state could be target joint angles, COM trajectory, etc. An optimal control problem consists of a rule governing the evolution of the future state as a function of the current state and current control signal, and a cost function to be minimized. Since it may be desirable to keep control signals small, for instance to minimize work, the cost may be a function of u as well as x. As the future state only depends on current information, and not the history of the system, a control policy which is optimal over some time interval $[t_0...t_2]$ is also optimal over a truncated interval $[t_1...t_2]$ where $t_0 < t_1 < t_2$. Hence, optimal control problems can often be tackled by working backwards progressively from the end of the interval, a so called dynamic programming approach. Variational analysis of optimal control problems leads to the Hamilton-Jacobi-Bellman (HJB) equation, a nonlinear partial differential equation. The solution of the HJB equation is the optimal value function, which gives the minimum remaining cost to the end of the time interval as a function of x. An optimal policy follows the gradient of the optimal value function.

An important special case of optimal control problem is the linear quadratic regulator (LQR), for which the rule governing the evolution of the state is a linear function of x and u, and the cost function is a weighted sum of quadratic functions in x and u. The LQR is analytically tractable and determination of optimal policies can be reduced to solution of an equation known as the Riccati equation. Methods also exist to tackle linear quadratic problems in the presence of Gaussian noise (LQG). Since analytical approaches are available, LQR based controllers can generally be evaluated with relatively little computation, allowing good performance. Some approaches to less tractable optimal control problems are discussed by Tedrake (2004).

A general problem with optimal control formulations for physically simulated characters is that the results may be highly dependent on the values chosen for the weights which express the relative importance of minimizing various components of x and u. Often these values must be chosen arbitrarily, based on physical insight, or by trial and error. Optimal control also generally copes poorly with discontinuities in dynamics due to contacts.

Brotman & Netravali (1988) use an optimal control approach to interpolate between keyframes in a physically valid manner, minimizing a cost comprising control energy and trajectory smoothness. They find that the results appear more natural and use less control energy compared to other interpolation methods. Such a method could be used as the basis of a motion editing approach.

Tedrake (2004) discusses some approaches to analytically intractable optimal control problems based on reinforcement learning of the optimal policy or optimal cost function. The approach is applied to a bipedal robot based on a passive walker with the addition of a small number of actuators. For this low dimensional system, a minimal walk is achieved after around one minute of online learning, and a robust walk after around 1000 steps.

Kwon & Hodgins (2010) use an LQR approach to determine optimal control for an inverted pendulum on cart model to generate a COM trajectory for a desired speed and heading which is similar to a reference running motion. The trajectory of the cart is then used to generate footstep locations and from these a full body trajectory is determined. From a single example motion, their method is able to generate running with various speeds and headings, and is able to recover from moderate disturbances. However, the recovery is achieved by artificial control forces on the cart rather than through foot placement, and appears unnatural. Performance is around half real-time.

Muico et al. (2009) find that LQR based controllers are unreliable for high dimensional characters, and propose an extended nonlinear quadratic regulator (NQR) formulation to deal with nonlinearities in the dynamics. They also treat the contact forces as part of the control policy, which allows planning over multiple footsteps. An LCP solver then calculates joint torques to correct discrepancies between the predicted contact forces and those actually obtained in the simulation. They demonstrate greater robustness and agile control such as sharp turns using NQR. Computation requires 1-10ms per frame depending on the complexity of contacts, and is dominated by the LCP solver, although it is noted that the solver used is unnecessarily precise.

Under certain conditions on the control cost and rule governing state evolution,

the HJB equation can be made linear by a change of variables. This allows linear superposition of existing controllers which are optimal for particular tasks, to generate controllers suitable for intermediate tasks, as demonstrated by da Silva et al. (2009). They form combinations of existing controllers with time varying weights determined according to their suitability for achieving the desired task from the current state. They demonstate examples including accurate control of entry angle for a variety of initial conditions for diving based on just two example controllers, and walking controllers able to adjust step lengths or cope with level or sloping terrain. Controllers combined in this manner achieve goals more effectively and with lower control cost than other approaches such as linear blending. Limitations include the requirement that individual controllers share a common optimal control formulation, and must operate over the same time interval. Performance is not explicitly stated, but the linear superposition approach is implied to be fast.

Muico et al. (2011) also make use of the linearizability of the HJB equation to track multiple controllers and they relax some of the limitations noted by da Silva et al. (2009) by providing a novel framework to blend or transition between controllers at arbitrary times based on the system state. Discontinuities in dynamics caused by contacts are problematic for the optimal control framework, but they reduce the severity of this difficulty in a similar fashion to their previous work (Muico et al. 2009) by incorporating the contact forces into the control policy and treating them as if they were actuation variables. By tracking multiple controllers, they are able to improve robustness. The authors also state that the push recovery is less stiff and unnatural than other control approaches, although they note that this is difficult to quantify.

2.5.5 Handcrafted Controllers

This section discusses controllers which use handcrafted models of biological mechanisms such as balance.

This kind of controller often uses proportional-derivative (PD) control to drive joints towards desired poses or to track reference motion. In PD control, actuation for each joint is given by a linear combination of the errors in the position and velocity of its corresponding DOF, scaled by appropriate gains. Typical responses of a PD controller are shown in Figure 2.5.2. If the position gain k_p and velocity gain k_d are related by $k_d = 2\sqrt{k_p}$, the dynamics are critically damped. Smaller or larger values of k_p relative to k_d lead to over-damping and under-damping respectively. Selecting good values for the position and velocity gains is the major difficulty in PD control. Gains which are too high can result in overly stiff motion, which lacks

responsiveness to disturbances, and can cause problems with overshooting target poses or oscillation. Biological control generally exhibits low stiffness, but gains which are too low can lead to very loose motion which appears unnatural and fails to reach target poses. Various heuristics can help to guide choice of gain values. For instance, selecting gains to obtain critical damping is often a reasonable starting point. van de Panne & Fiume (1993) suggest that sensible initial values can be determined by simple calculations, for example a bent leg should be able to produce an actuation comparable to half the character's weight. Zordan & Hodgins (2002) suggest that gains should be scaled for each joint based on the inertia of the attached chain of body links. Zordan & Hodgins (2002) also use a temporary reduction in gains following impacts to provide responsiveness before ramping back to original values. These principles can give a good starting point, but fine tuning of gains is often necessary, either by trial and error or optimization (see section 2.5.6). Gain values may also be highly dependent on morphology and require fresh tuning for different characters. A major advantage of PD control is that is conceptually simple and computationally inexpensive.

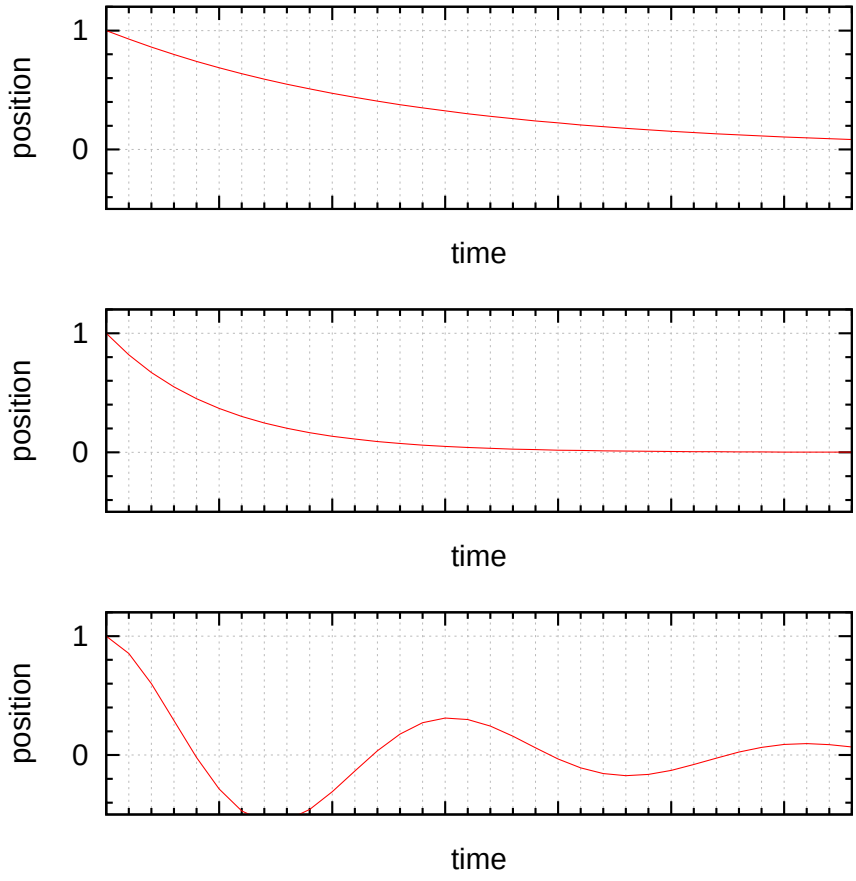

Figure 2.5.2: Typical Responses for Proportional-Derivative Control, for Over-Damped (Top), Critically Damped (Middle) and Under-Damped (Bottom) Conditions

Antagonist control is a similar representation, in which a pair of springs is associated

with each DOF, and control is achieved by varying the two spring stiffnesses. This formulation is entirely equivalent to PD control, but it has the advantages that it is more conceptually similar to actuation in real animals, the parameters are more intuitive, and it separates control of position and stiffness (Neff & Fiume 2002).

An integral component is sometimes incoporated to include feedback on the history of the dynamics, leading to proportional-integral-derivative (PID) control. This can have the advantage of detecting that applied actuation is not sufficient to achieve target poses and adjusting accordingly, but the integral is invalidated by discontinuities in motion.

Allen & Faloutsos (2012) describe a number of misconceptions regarding PD control, in particular regarding its asymptotic behaviour, the time to reach the target position, and the effect of a target velocity term on the final position.

In work by Allen et al. (2011), PD control parameters are determined in order to precisely interpolate target states in a critically damped fashion, matching both position and velocity targets. The method is applied to a single-DOF robot arm with rapidly changing target positions during a ball deflecting task. The calculated PD parameters give much closer tracking of desired position and velocity compared to a hand-tuned controller, and the resulting motion appears less robotic due to a more natural compliance.

Handcrafted controllers often use a finite state machine (FSM), consisting of a small set of states corresponding to phases of locomotion such as leg swing, toe-off or the flight phase for running. Transitions between states may occur on a timed basis, or may be triggered by events such as heel strike. Some methods use an associated pose control graph, with target poses for each state. Modifying target poses can be a straightforward way to generate new motions (Coros et al. 2010), but control over motion style is difficult. Setting target positions may be relatively easy, but target velocities are much less intuitive. For this reason, target velocities may be set to zero, requiring the use of exaggerated target poses which are never reached. In general, using pose control graphs is uncomplicated, but less detailed and natural than tracking reference motion, with control over style limited and difficult.

Raibert & Hodgins (1991) simulate running motions for bipeds and quadrupeds using PD control. Elastic mechanisms in the legs are modelled, and energy injected or removed to produce the vertical oscillations observed in real running motions. Speed and balance are controlled using foot placement based on an IPM. Balance is achieved through PD control of the trunk attitude.

Hodgins et al. (1995) produce simulated running, cycling and vaulting motions using a state machine to select control laws based on the motion phase. PD control is used

to drive joint angles towards targets computed by the control laws. Remaining DOFs not directly associated with the motion are used for stabilization. The control laws are hand-tuned using force plate data and muscle activation data from biomechanics, and motion capture. Their method is stated to be most suitable for highly dynamic motions, which are strongly constrained by dynamics.

Zordan & Hodgins (2002) use PD control to track motion capture data for simulation of boxing and table tennis. IK and timewarping are used to modify the motion capture data, for example to generate new table tennis swings. The tracking is augmented with balance control achieved by computing additional torques to drive the COM position over the centre of the support polygon. PD gains are temporarily lowered when impacts are detected, increasing responsiveness.

SIMBICON (simple biped control) is a successful and often cited example of a handcrafted controller (Yin et al. 2007). It uses PD control with a pose control graph containing four states for walking or two states for running motions. Alternatively, it can use tracking of a reference motion. A robust balance mechanism is achieved by expressing the target angles for the torso and swing hip in world space, and controlling the swing hip target angle based on feedback from the COM trajectory. Feedback error learning is used to learn largely feed-forward control, eliminating the need for large PD gains. The controller is able to produce a variety of motions, including walking in all directions, running and hopping, and impressive robustness to pushes is demonstrated. There are limitations, including robotic gaits, and poor reproduction of realistic ankle motion and toe-off behaviour. The controller runs in real-time on standard hardware.

Coros et al. (2010) describe another method based on PD tracking of a pose control graph, with the aim of allowing high flexibility and straightforward authoring of new motions. In a similar fashion to SIMBICON, some joint angles are expressed in world space to include information about orientation. Balance is achieved through foot placement controlled using an IPM. High level control of locomotion speed is achieved by using virtual forces to accelerate or decelerate the COM. The virtual forces are mapped to joint torques via the Jacobian of the system. Jacobian transpose control is also used to apply compensation for estimated gravity forces to allow low-gain PD tracking. The controller is designed to be insensitive to changes in gait, task and morphology, and avoids tuning of parameters by using critically damped PD control, with gains scaled based on mass for different characters. Generalization of the method across character morphology and various tasks including picking up and carrying heavy objects, and climbing over and ducking obstacles is demonstrated. The controller is not suitable for highly dynamic motions and, since a pose control graph method is used, the results do not appear as natural as motion capture. Real-time performance is achieved using standard hardware.

Jain & Liu (2011*a*) consider a SIMBICON controller with spring forces at contact points used to model the effect of deformable bodies. They find that this approach allows greater robustness and realism compared to using rigid bodies, which tend to result in intermittent contacts and sporadic pressure distributions. They also consider some other approaches such as segmenting the foot into multiple rigid bodies and penalty methods. However, they find that segmenting the foot introduces problems with mass ratios and additional controller complexity, while penalty methods with inappropriate stiffness may result in penetration or bouncing.

Handcrafted controllers are generally highly task specific, and constructing more complex motion spaces with many tasks requires transitioning between controllers. The success of a transition depends on the two controllers and the controller phase at transition, and determining which transitions can be made stably is a difficult problem. Faloutsos et al. (2001) address this problem by using support vector machines to classify stable entry conditions for controllers.

Although handcrafted controllers have produced some impressive results, there are a number of difficulties with this approach. Manually designing controllers is a skilled and time-consuming task, and controllers tend to suffer from high specificity in terms of task, character morphology and environment. As controllers tend to be highly gait specific, providing agile and responsive control involves transitions between controllers and is a difficult problem. PD control generally handles each joint individually, which can lead to poor simulation of the highly co-ordinated motion seen in nature. The fact that poses are targets only and may not be reached can hamper precise control.

2.5.6 Offline Optimization of Controller Parameters

As discussed earlier, many types of controller include parameters, such as PD gains, weights in cost functions of optimal control formulations, and so on, which must be specified. In some cases, their values may be guided by physical insight, but often their effect is not very intuitive. They are also often sensitive to variation, requiring retuning for changes in character morphology or task. One common way to address these problems is to perform offline optimization to find parameter values that perform well for some desired task.

The objective functions to be optimized are typically high dimensional, nonlinear and discontinuous with many local minima. These factors make it difficult to apply many standard optimization techniques, although stochastic methods can be effective. Covariance matrix adaptation (CMA), illustrated in Figure 2.5.3, is a stochastic approach which is frequently used (Hansen 2006). In CMA, search points

are randomly sampled and evaluated from a multivariate normal distribution which represents the current estimate of the optimum. An "elite" subset of these samples with the highest values of the objective function is used to calculate an improved estimate of the mean and covariance, and the process is iterated until some convergence criterion is achieved. Step-size adjustment arises naturally out of the variance estimation, allowing fast progress but preventing premature convergence. The method also has the advantage that it requires only function evalutions, not derivatives. Problems of the complexity involved in controller optimization can take many hours or days to converge.

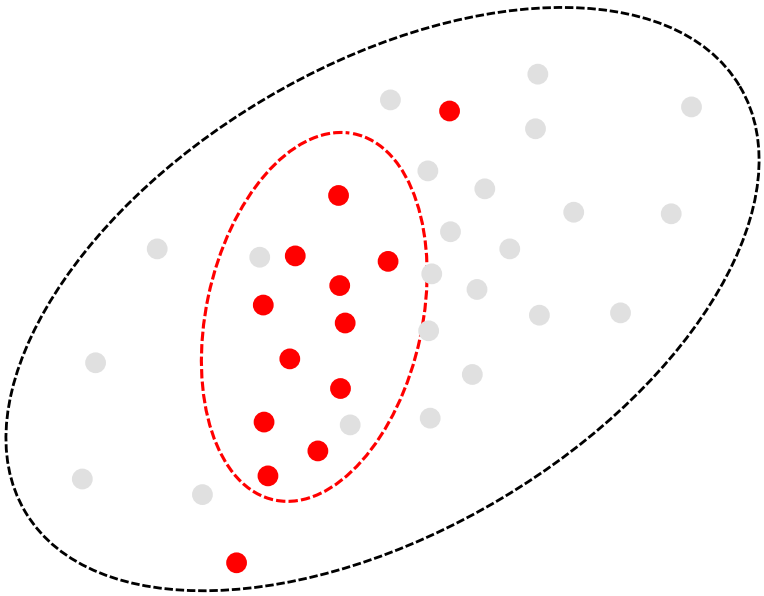

Figure 2.5.3: Illustration of Covariance Matrix Adaptation Method - Elite Samples (Shown in Red) from One Iteration Determine the Next Iteration's Sampling Region (Red Ellipse)

Sharon & van de Panne (2005) look at low dimensional planar biped locomotion using a set of nodes in the space describing the system, with control for a particular state determined by the closest node. The nodes thus form a Voronoi decomposition of the state space, and control behaves similarly to a FSM, with traversing a cell boundary corresponding to a state transition. The positions of the nodes and their outputs are optimized to give motions close to a provided reference sequence. The reference motion can be used to give a good initialization for the optimization problem, and the cost function is carefully chosen to give a more tractable search space, with the result that a relatively unsophisticated greedy hill-climbing method can be used. The technique is demonstrated for a variety of motion styles, and for uphill climbing and robustness to disturbances.

Yin et al. (2008) look at the use of offline optimization to improve flexibility of existing controllers in an automated fashion. They use continuation methods, an approach to solving for different scenarios by varying some parameter of the system

such as step height or coefficient of friction in small increments and optimizing based on the previous solution. This generates a continuous range of controllers which can be interpolated to tackle intermediate problems. They consider a variety of optimization methods based on gradient descent or stochastic approaches. Robust adaptation is demonstrated for diverse variations including tall steps, steep inclines, pushing heavy objects, and walking on ice.

Wang et al. (2009) use CMA to optimize a slightly modified version of the SIMBICON controller (Yin et al. 2007) to improve naturalness. They note some discrepancies between the original SIMBICON and real locomotion, including unnaturally high torques in the hips and knees rather than the ankles, and lack of realistic toe-off behaviour. To address these issues, extra bones are added in the feet, and control of the ankles is handled in world space rather than local joint space, to ensure good ground clearance. Transitions in the finite state machine are determined by motion heuristics rather than the time-based approach used originally. They include objective terms in the optimization to encourage realistic walking characteristics such as good foot clearance, low overall angular momentum, low head motion and largely passive knee joints. There are also optional objective terms to control walking speed and step length. They demonstrate significant improvements in realism over the original SIMBICON, including closer agreement with measured leg joint trajectories. Robustness is somewhat reduced, since it is not explicitly optimized for, and SIMBICON uses some unnatural mechanisms to maintain stability. Optimization involves up to 3000 iterations of CMA, requiring about 180 hours of CPU time.

In work by Wang et al. (2010), controllers are optimized for various sources of uncertainty, including unexpected forces and motor noise. They also optimize for the influence of user input and controller transitions not known in advance, important considerations for responsive control. A Monte Carlo approach is used to evaluate controllers in the presence of uncertainty, and optimization is performed with CMA. To deal with convergence difficulties, controllers are optimized in incremental steps of gradually increasing noise starting from a baseline with a similar method to that used by Yin et al. (2008). They are able to demonstrate relatively agile user control, increased tolerance to pushes and robust walking on narrow beams and slippery surfaces. Realistic behaviour such as taking short steps and using arms for balance on narrow beams arises automatically.

Wang et al. (2012) optimize a controller based on biologically inspired actuation and building on the positive force reflex ideas suggested by Geyer & Herr (2010). They use torques generated by Hill-type simulated musculotendon units (MTU) (Zajac et al. 1989), which model the dependence of actuation response on muscle length and contraction velocity, and they also model the dependence of joint torques on moment

arm lengths. 8 MTUs are used in each leg, corresponding to the major human leg muscles, including biarticular muscles (muscles which span two joints), such as the hamstrings. No reference motion is used, and control is achieved with a combination of SIMBICON style PD control and biologically motivated reflexes such as positive force feedback. The MTUs introduce many additional parameters, and the controller has over 100 tunable values, which are optimized using CMA to minimize metabolic energy expenditure per unit distance. Walking and running are both demonstrated by varying the target velocity. The controller produces impressively realistic results, and by modifying parameters it is also possible to reproduce behaviour observed in patients with conditions such as weak hamstrings. The controller uses a physics simulation update rate of 2400Hz, and is stated to run in real-time once optimized.

Kwon & Hodgins (2010) use CMA to optimize adjustments to footstep locations and knee torques generated by IPC trajectory planning (see Section 2.5.4) in order to improve motion quality and provide better tracking of reference motion.

In work by Wu & Popović (2010), biped locomotion control is formulated in terms of three end-effectors for the feet and upper body. A high level planner determines a desired end-effector trajectory for each footstep. A low level controller solves a quadratic program to calculate torques to achieve the planned trajectory. Offline optimization of the planner parameters is then performed using CMA to minimize energy cost and deviations from desired COM trajectory. It is noteworthy that balance is not explicitly modelled, but is achieved through the optimization by learning how to modify the end-effector trajectories to influence the COM in the desired fashion. Helper forces are allowed in the optimization, but penalized, and designed to be phased out as optimization progresses. The planner samples terrain height values along the path of the foot and adjusts a height offset accordingly, allowing the controller to avoid the unnaturally high ground clearances seen in terrain-blind approaches. Impressive results are shown, including natural and robust locomotion on uneven terrain, and highly agile transitions such as turning 180 ° in two steps. Because control is formulated in terms of end-effectors only, the approach can easily be applied to different character morphologies. The controller operates in real-time on standard hardware, although it uses 2048 simulation time steps per second.

Ding et al. (2012) use CMA to learn linear feedback policies to stabilize open loop control for tasks including balancing, walking and running. They reduce the number of parameters by decomposing the feedback matrix into low dimensional state and actuation projection matrices, which encourage co-ordinated sensing and control. Matrix elements are also encouraged to be zero to increase sparsity further. The reduced order feedback parameters are optimized using CMA by evaluating policy rollouts. Their approach discovers low-dimensional feedback policies of comparable effectiveness to hand-tuned controllers such as SIMBICON, and shows that ground

reaction forces and centres of pressure can be useful for control. In some cases, lower dimensional feedback policies achieve better fitness, as for higher dimensional models there may be problems with convergence or local minima. Some of the models take hundreds of hours to optimize.

Liu et al. (2012) use a similar method to learn feedback control for running and a set of obstacle clearing manoeuvres. Starting from a single motion capture sequence for each task, open loop controllers are generated using the method described by Liu et al. (2010) (see Section 2.5.8), with learned feedback as used by Ding et al. (2012). These specific controllers are then parameterized using a continuation method. For running, new optimizations are performed for slightly different conditions of speed and turning direction, and fitting with radial basis functions is used to allow interpolation. Similarly, the obstacle clearing skills are parameterized with respect to features such as obstacle height, with parameterized optimal approach distances and velocities also determined. Further optimization and control blending are used to achieve robust transitions between running and obstacle skills. Simple footstep planning is used to get as close as possible to the ideal distance and velocity values when approaching each obstacle. The results demonstrate impressive progress in robustly transitioning between highly dynamics tasks, although appear to be hampered by the limited extent of the turning parameterization. Obstacle clearing success ranges from 83-90%. Simulation is performed using Open Dynamics Engine (ODE) with a time step of 0.5ms. Each optimization requires several hours, with the 2D parameterization of the running skill requiring 11 hours.

In the work of Geijtenbeek et al. (2013), both control parameters and muscle routing are optimized using CMA. Muscles paths are represented as sets of line segments connecting attachment points via intermediate points, with the attachment and intermediate points free to be modified during optimization. A Hill-type model of muscle dynamics similar to that employed by Wang et al. (2012) is used. More robust and natural results are obtained using this approach than when the muscle routing optimization is disabled. Simulation is performed using ODE with a time step of 0.3ms, with a spring-damper contact model.

2.5.7 Online Optimization

In this section, some methods which use an optimization approach but operate in real-time or close to real-time are discussed.

In online optimization, control is often formulated in terms of quadratic objective functions with linear constraints, resulting in quadratic programming (QP) problems for which efficient solvers are available. An overview of a typical online optimization

approach is shown in Figure 2.5.4. In general, these techniques have the advantage that optimization considers the character DOFs simultaneously rather than individually, resulting in more co-ordinated motion and allowing more faithful tracking of reference motions. However, defining suitable optimization objectives can be difficult, and objectives can interfere with each other (Abe et al. 2007). These methods are also very demanding computationally and, although described as real-time in the literature, would be very hard to implement using the limited resources typically available in games.

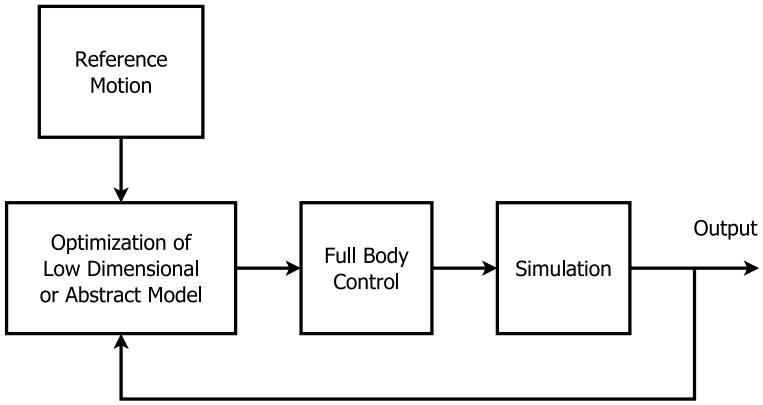

Figure 2.5.4: Overview of Typical Online Optimization Method

Abe et al. (2007) use online optimization to solve for joint torques based on a set of quadratic objectives describing goals such as tracking, balance and reaching. Balance control is formulated in terms of driving the COM position over the centre of the support region. The solver may optimize the objectives in priority order, or combine them as a weighted sum. The prioritization approach is found to give less realistic results due to interference between objectives. Control is demonstrated for a variety of reaching and robust balancing tasks. Since the objectives are not morphology dependent, the control scheme can easily be applied to different characters. The controller runs at 30 fps on standard hardware.

Planning over long intervals is computationally intractable in real-time, and computed trajectories are quickly invalidated by disturbances. da Silva et al. (2008*b*) address these problems by using short-horizon planning, and updating plans regularly. Planning is achieved by using a linearized model of dynamics as a constraint in a QP, which is solved to compute torques required to track reference motion over a short interval. The predictive control is augmented with feedback from a low gain PD controller, to compensate for errors and deal with unexpected disturbances. Their method also employs a balance mechanism based on control of the swing hip, similar to SIMBICON. The controller is able to successfully adapt reference motion to new environments, such as inclines. Motion appears somewhat robotic, a result attributed to the greedy nature of the short-horizon planning. The controller operates in real-time.

da Silva et al. (2008*a*) combine tracking of reference motion and balance control. Balance control is modelled using an optimal control formulation, with a quadratic cost function. The balance controller uses a reduced 3-link model (2 legs and trunk) to improve performance and to suppress DOFs which have little significance in locomotion. A linear approximation to the dynamics is computed offline for each contact state of the reference motion, allowing the optimal control problem to be cast as an analytically solvable LQR, which can be evaluated for the reduced model efficiently. A style controller tracks the reference motion, and style and balance terms are combined in a QP. Faithful tracking of reference motion over difficult terrain including slopes, steps and seesaws is demonstrated.

Macchietto et al. (2009) consider real-time balance and tracking control for a standing character. Unlike many other methods which address only the COM, both COM and COP (or equivalently both linear and angular momentum) are controlled, based on evidence from biomechanics that angular momentum is tightly controlled in nature. Control is achieved by calculating ideal joint accelerations through online optimization of a quadratic goal comprising terms representing linear balance, angular balance, and reference tracking. A penalty method is used for contacts with the result that the quadratic problem can be solved in a single step, as opposed to a constraint based approach which would require an iterative solver. Torques are then computed from the ideal accelerations using an inverse dynamics solver. The addition of angular momentum control leads to co-ordinated motion which is richer and more robust, with natural behaviour such as windmilling arising automatically. Since control is based on high level descriptions of the character such as COM and COP, it is easily adaptable to different morphologies. The controller operates in real-time.

Wu & Zordan (2010) use a similar approach to that described by Macchietto et al. (2009), but add the ability to take deliberate steps, with high level control of stepping location and duration based on the character's momentum state. Based on research on natural trajectory formation (Abend et al. 1982), a target path for the swing foot is given by a straight trajectory with Gaussian velocity profile. A target trajectory for the COM is also defined, using a quadratic curve, consistent with motion capture data and with the idea of the swing phase as a controlled fall, similar to the IPM. Target joint angles are then determined using IK, and combined with balance goals in a quadratic problem, which can be solved in real-time. Realistic reactive stepping is demonstrated and, as with the work of Macchietto et al. (2009), the control can handle different morphologies easily.

Mordatch et al. (2010) use online optimization to deal with planning for difficult locomotion tasks such as moving on uneven terrain. Planning is conducted over the course of two steps using a closed form approximation to a spring loaded inverted

pendulum. Six phases of motion are considered, a double stance, single stance and flight phase for each leg. This allows different gaits to be represented in the same framework, with some phases having zero duration in certain gaits, for instance walking has no flight phase. The planning objectives include user control terms to influence step size and heading. The planning optimization problem is solved online using CMA, initialized using information from previous planning steps to improve performance. Full body torques are determined using a QP, with objectives to follow the low dimensional plan and maintain balance. Robust response to pushes and traversal of difficult terrain including slopes and variously sized gaps is shown, although the motion is somewhat robotic and overly stiff. Performance is around 15% of real-time, or 55% with reduced planning frequency.

de Lasa et al. (2010) accomplish locomotion through control of high-level motion features including angular momentum and COM and end-effector trajectories. Reference values for these features are determined using state machines, and quadratic objectives describing their behaviour and including user input are optimized in a prioritized fashion. Precise jumping and walking in a variety of styles is demonstrated. As control is based on high-level motion features, parameters are intuitive, and controllers can easily cope with different characters. Performance is around 50-100% real-time.

In work by Ye & Liu (2010), an optimal control policy is calculated offline for a reference motion using differential dynamic programming with an abstract dynamics model based on linear and angular momenta and COM position. Approximating derivatives about the optimal policy gives a linear feedback policy in terms of the abstract model state, time to complete the motion and the final state. The ability to alter completion time gives more robustness to disturbances, and changing the final state constraint allows environmental adaptations such as step height. A quadratic problem is then solved to calculate full body motion consistent with the abstract state. Simulation is performed at 120Hz. Performance is about 15% of real-time for a 42 DOF character.

Jain & Liu (2011*b*) perform long horizon planning, using modal analysis to simplify computation by allowing uncoupled modes to be optimized separately (see also the work of Kry et al. (2009), discussed in Section 2.5.8). First a quadratic problem is solved to compute contact forces to achieve bulk motion as consistent as possible with the target motion. Then each mode is optimized using the already calculated contact forces. Planning is performed for low frequency modes over an interval of 10-15 frames, with shorter horizon planning for high frequency modes. For a 42 DOF character, performance is around 3-10% real-time, with computing the interaction between contact forces and bulk motion being the bottleneck.

Tassa et al. (2012) solve a linear quadratic regulation problem at each time step to achieve control. Solutions from previous time steps are used to warm start the solution. A smooth approximate contact model allows contacts to be differentiated. The resulting controller is able to demonstrate getting up from arbitrary poses, and recovery from large disturbances. For a 22 DOF character, performance is around 15% of real-time.

Brown et al. (2013) look at rotational movements such as rolling, which are infrequently studied. As well as angular momentum and angular velocity, they also find that the integrated angular velocity, which they refer to as angular excursion, is effective as a high-level control objective. Control is achieved using a quadratic problem to optimize high-level linear and rotational objectives along with pose tracking. Rotational behaviours frequently involve many potential contacts. To determine which should be used for control, the optimization is first run with all contacts, then those exerting little force or torque about the COM are excluded. Flexibility of the pose tracking is improved by blending between two target poses. Optimization and simulation are both performed at 60Hz. The single-threaded implementation runs at 20-40% real-time.

Al Borno et al. (2014) also consider rotational movements such as cartwheels, dives and flips. A quadratic problem is solved at each time step to optimize contacts, COM and end effector trajectories and angular momentum. Control is parameterised using a time-invariant phase variable based on the angle of the body, resulting in greater robustness to disturbances compared to a time-based parameterization. The control and simulation both run at 1000Hz, with single-threaded performance around 50-100% real-time.

Han et al. (2014) solve a series of optimal control problems over small time windows covering two footsteps. A low dimensional nonlinear dynamics model is used, based on overall momentum change and foot motion. Optimization is prioritized, with the highest priority level solving for foot motions, allowing foot contact repositions for robustness and uneven terrain. A second level of priority solves for linear and angular momentum, energy and reference tracking objectives. Output motion is generated at 200 frames per second, with the optimal control policy updated as often as performance allows, initialized using previous solutions. The dynamics formulation allows closed-form derivatives, allowing real-time performance.

2.5.8 Other Methods

This section describes some methods which do not fit neatly into the categories already discussed.

Various methods employ control strategies based on the behaviour of systems in phase space. Cyclic motions such as walking can be represented by closed phase space trajectories known as limit cycles. Laszlo et al. (1996) formulate control by linearizing dynamics about a desired limit cycle and driving disturbed states back to the stable trajectory. Another related concept is a Poincaré map or return map, which describes how the state evolves in subsequent orbits of periodic motion, with fixed points on the map corresponding to stable trajectories. Tedrake (2004) uses return map analysis to generate stable walking for low dimensional bipedal robots based on passive walkers.

Pratt et al. (2006) analyse the dynamics of an inverted pendulum plus flywheel model and derive expressions for the region into which a biped must step to come to a complete halt, which they describe as the capture region. The model is applied to a low dimensional representation of a biped, which is able to stop by using angular momentum based strategies such as lunging.

Coros et al. (2008) use a combination of offline and online methods to accomplish locomotion planning on highly constrained terrain. For a given constrained locomotion task, solutions for a large number of randomly generated similar tasks are computed using offline optimization. A non-parametric regression is then performed on the population of examples to produce a model of the character's state after stepping based on its current state and selected action. To simplify the analysis, a low dimensional model of the state is manually defined, and a low dimensional action space is determined using PCA on the example motions. The resulting stepping model is used online to plan over two step intervals, with replanning after each step. Planning may be achieved by systematically sampling the space of feasible actions for highly constrained motions, or by random sampling for less constrained problems. Navigation of difficult terrain including anticipation and variation of step size is demonstrated, although the movement is somewhat lacking in naturalness. The controller runs in real-time or close to real-time, depending on the action space sampling method.

Kry et al. (2009) consider motion in terms of a character's natural oscillatory modes (see also the work by Jain & Liu (2011*b*) discussed in Section 2.5.7). They focus on low frequency modes which they argue are most likely to correspond to natural energy efficient motion. Simple loop-shaped and worm-like creatures are physically simulated by activating relevant modes with particular amplitudes and phases, but a lack of balance control prevents simulation of more complex creatures. Kinematic motions for a dog are demonstrated by combining modes relevant to locomotion. Up to six modes are used.

Liu et al. (2010) achieve open-loop control by randomly generating PD joint targets

within specified sampling windows. At each sampling step, a large number (1400 in the examples considered in the paper) of possible future states are generated using simulation. The states are evaluating using an objective function based on tracking a reference motion. To retain diversity and avoid shortsighted control, a smaller number of states (200) are kept as the basis of future sampling, with retention biased to better values of the objective function. Control over a series of sampling steps can then be reconstructed by backtracking from the final state. The method is illustrated in Figure 2.5.5. The approach is used to successfully reproduce various motions including walking, running and contact-rich rolling. It can also be used to transform reference motions to different environmental conditions, such as uneven terrain or low friction, and to retarget motion to other characters. Simulation is performed with 0.5ms time steps, with control sampling at 0.1s intervals. For 1400 sampled motions per sampling step, reconstructing motions requires 1-3 minutes using 80 cores.

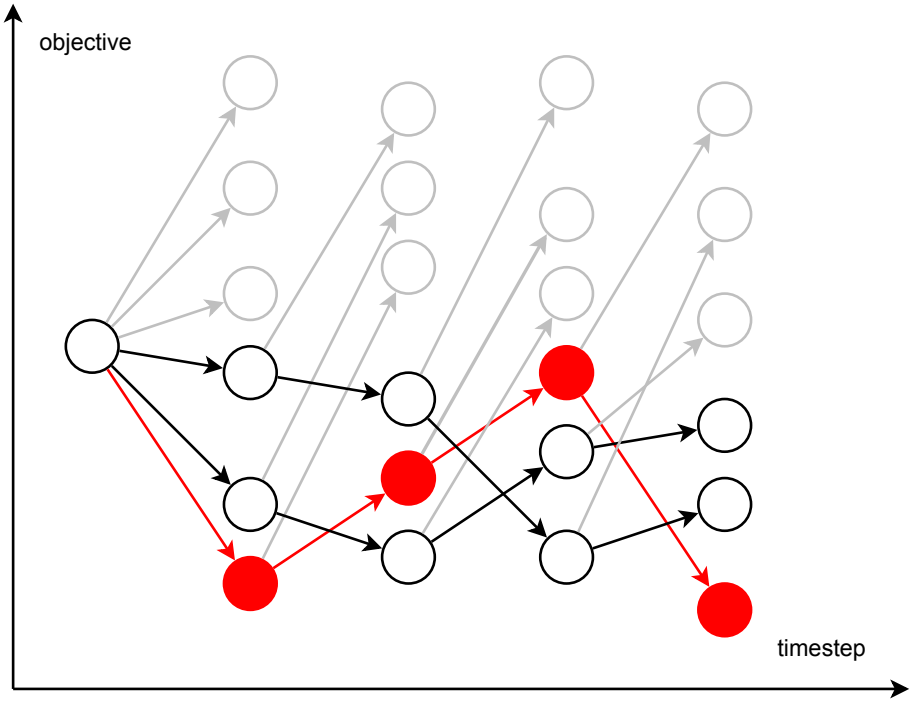

Figure 2.5.5: Sampling-based Approach as Used by Liu et al. (2010)

2.6 Conclusion

This section describes recent trends and possible future directions in physics-based character locomotion and discusses some open questions.

As shown in Figure 2.6.1, the field is currently enjoying an active period. Activity appears to have declined a little since a particularly high period around 2010, but remains high.

Examining the individual categories in Figure 2.6.1, some trends are suggested, although care should be employed since the category sample sizes are relatively small, papers may been missed during the literature search, and there may be some effects due to the difficulty of classifying research which incorporates ideas from more than one category.

It appears that handcrafted controllers have become uncommon in the last few years as tuning of multiple parameters is difficult and time consuming. At the same time, offline parameter optimization methods have become more prevalent.

Spacetime optimization continues to be a useful technique for offline motion editing and synthesis, and recent work to improve handling of contacts such as that by Mordatch et al. (2012) and Posa et al. (2014) extends its usefulness by increasing the flexibility and robustness of search space exploration.

There has been some interesting recent work in the area of neural networks, such as by Liu et al. (2013), but these methods remain difficult to apply to characters with many DOF.

There has been a lot of recent activity in the area of online optimization. These approaches can generate very impressive results, and with recent approaches such as applying optimization to simplified low dimensional models, performance is approaching real-time. With further developments in performance, this could be a very promising area for future work.

Another recent trend has been the use of more biologically influenced actuation, for example as employed by Wang et al. (2012) and Mordatch et al. (2013). These approaches increase complexity, but give impressively natural results without the need for reference motion.

Although quite natural results have been achieved using biologically inspired actuation and objective functions, it remains unclear whether certain aspects of motion style such as emotional state and fatigue can be represented without using reference motions.

There has been relatively little work on rigorously assessing the naturalness of generated motion. Although some methods such as comparing joint angles and moments with observed human trajectories have been used (Wang et al. 2012), many papers rely on visually assessing the plausibility of output motion, making it difficult to compare different approaches.

Although there has been a lot of progress with coping with unexpected forces and changes in terrain, there is still some way to go to achieve a flexible controller able

to transition smoothly between different tasks and adapt robustly to environmental changes. For interactive control, balancing realism with prompt responsiveness to unexpected user input remains challenging, although this issue is not specific to physics-based methods.

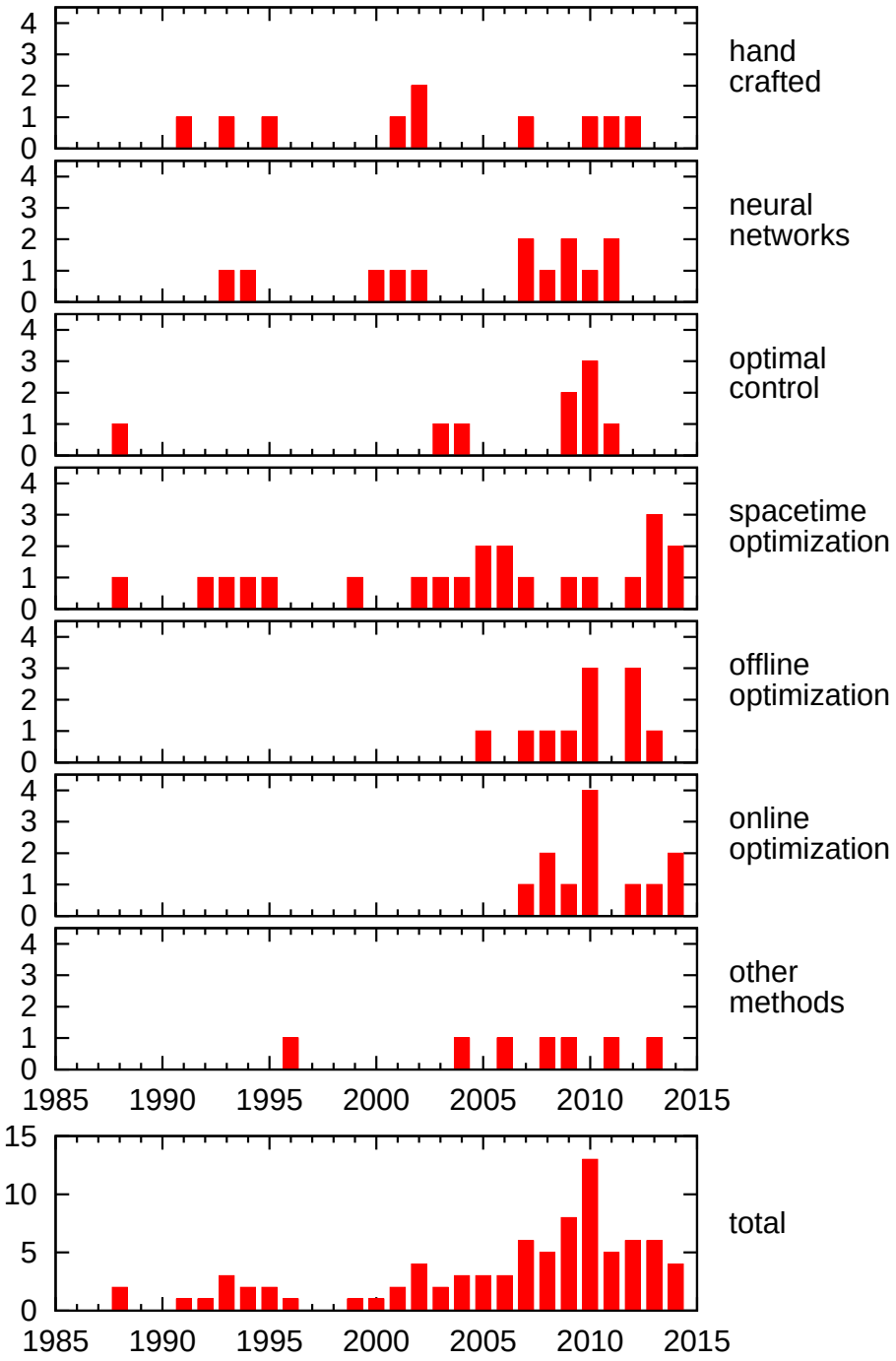

Figure 2.6.1: Papers Cited in Section 2.5 Broken Down by Year and Category

Chapter 3

Preliminary Implementation and Results

3.1 Introduction

As discussed in Chapter 2, there is a large body of existing work on physics-based character control, drawing on many areas including animation, biomechanics and robotic control. This project aims to build on the existing work and address some questions concerning real-time applications using commercial physics engines. Much of the existing work is not designed to run in real-time and even methods described in the literature as real-time often require too much CPU time to be immediately applicable to applications such as games where animation control must run alongside many other tasks. Various problems need to be considered in order to achieve real-time performance. In particular, it may be necessary to use longer time steps in the numerical integration stage of the physics engine than are often used in existing work, making control more difficult.

3.2 SIMBICON Type Controller Implementation

SIMBICON (Yin et al. 2007) was chosen as the basis for initial experimentation as, although the mechanics of the controller are relatively simple, it is quite effective and robust, and it is widely cited in existing work, with many researchers achieving good results with additional sophistication built on top of SIMBICON-style controllers (Yin et al. 2008, Coros et al. 2008, Wang et al. 2009, 2010, 2012). This section gives some more detail on the original SIMBICON paper and the development of a preliminary implementation integrated within NaturalMotion's animation engine,

morpheme.

The original SIMBICON method is a joint based PD control approach, with torques for each joint calculated based on the discrepancy between desired and simulated values of the joint angle and angular velocity, scaled by gain parameters k_p and k_d respectively.

$$\tau = k_p(\theta_d - \theta) + k_d(\dot{\theta}_d - \dot{\theta}) \tag{3.1}$$

In the original SIMBICON method, desired joint orientations are defined by a finite state machine with a small set of target poses corresponding to phases of the locomotion cycle, with transitions between phases triggered either by time or by heel strike events, and target angular velocities are set to zero.

For most of the joints, the PD control is applied in the joint's local space as defined relative to its parent. In order to include information about the vertical direction which is relevant for balance, control for the torso, swing hip and stance hip joints is applied in world space. The torques for the swing hip and torso, τ_B and τ_{torso}, are determined based on their desired orientations, and the torque for the stance hip, τ_A, is calculated to ensure that the net torque is zero.

$$\tau_A = -\tau_{torso} - \tau_B \tag{3.2}$$

A balance mechanism is provided by modifying the target orientation for the swing hip with feedback based on the horizontal position of the COM relative to the stance ankle, d, and the COM velocity, v.

$$\theta_d = \theta_{d0} + c_d d + c_v v \tag{3.3}$$

Source code for a standalone implementation of SIMBICON was obtained from the authors' website. This version used an update frequency of 2000Hz for the physics simulation, and was based on Open Dynamics Engine (ODE), which uses a matrix solver based approach to resolve constraints, which is highly accurate and robust but computationally expensive. Although real-time performance was achieved, the CPU utilization was very high, which would be unsuitable for applications such as games where character control must share resources with many other tasks. Experimentation with the source code suggested that it was possible to lower this physics update rate somewhat, but the controller became completely unstable at around 750Hz. Likewise, Giovanni & Yin (2011) implement generalized biped walk-

ing control (Coros et al. 2010), a similar controller to SIMBICON, in several different simulation engines and find that the controller fails for time steps above 1-2ms for all the simulators considered.

Initial work focused on developing an implementation of a SIMBICON-style controller integrated into morpheme, with computational requirements more suitable for real-time applications. The default male character rig and example walking motions from morpheme were used. The rig has 20 links with proportions and mass distribution suitable for representing a human, although some minor changes are made to improve simulation characteristics, such as avoiding large mass ratios in chains. Twist and swing based joint limits are used to allow a realistic range of motion. The character's feet are represented by rigid cuboids. The implementation was based on a modification of an existing morpheme physics node, which tracks target animations using a spring-damper based PhysX joint drive. A world space controller was added to calculate torques for the three joints associated with balance, and the existing joint drive was disabled for these joints. Performance was improved by scaling torques for the balance joints based on the moments of inertia of their attached chains. The SIMBICON balance feedback terms were also added to modify the target orientation for the swing hip. Rather than using a small set of target poses, desired orientations for joints were determined based on a reference walking animation, with the aim of improving the naturalness and level of control over motion style compared to the original SIMBICON, which often appears robotic.

For each frame, it is necessary to identify which leg is the swing leg, and which is the stance leg. This was initially determined by retrieving footfall timings from the event information associated with the reference animation (events such as heel strike are marked up in the animation engine for reasons such as avoiding blending artefacts), but this was not found to be satisfactory due to latency between the simulated and target poses. A more effective approach involving contact information retrieved from the physics engine was implemented. The contact information could also be used to reset the phase of the target animation at each footstep to avoid losing synchronization.

As one of the aims of this project is to investigate whether locomotion control can be brought more in line with the performance requirements of real-time applications, the development and experimentation with the initial morpheme implementation was performed with a physics simulation rate of 60Hz.

With the initial implementation, the character was able to take one step, but appeared to lack sufficient momentum to propel itself over the new pivot foot and tended to fall backwards. An alternative feedback mechanism based on ensuring the COM had sufficient energy to pivot was investigated, but was no more successful.

The implementation here used a reference motion rather than a FSM containing a small set of target poses, so may not be directly comparable to the work by Yin et al. (2007) but, as noted above, similar FSM based approaches have also been found to fail with high time steps (Giovanni & Yin 2011).

3.3 Foot Contacts

A major source of instability in the initial implementation was the interaction between the feet and the ground. One difficulty was with errors and latency in the tracking resulting in poor orientation of the foot prior to heel strike, leading to tripping. Another problem involved dealing with the sudden impulse associated with the ground impact and achieving a sufficiently stable foot plant to support the subsequent step. Ground contacts introduce large, discontinuous forces which increase the stiffness of the numerical integration in the physics simulation. At a physics rate of 60Hz, the uncertainties associated with large integration time steps make the behaviour of the simulation less predictable and smooth. Furthermore, at this simulation rate, the controller only has 2 or 3 physics updates after each heel strike to accomplish a stable stance with that foot ready to initiate stepping with the opposite foot.

An attempt was made to address the lack of stable foot plants by creating a constraint to lock the foot in position immediately after heel strike, and destroying it when the COM had moved a short distance in front of the pivot point. This approach was successful in producing a controller which could take many stable steps and reproduced the input animation relatively well (see accompanying video). However, it was not satisfactory as a long term solution as stability was achieved in a very artificial manner, with the character appearing to lean backwards on the step further than would be stable without the foot locking, and being dragged over the pivot foot rather than being pushed by the trailing leg.

3.4 Tracking Latency

Subsequent work aimed to achieve stable walking without the unnatural effects associated with applying constraints to the feet after heel strike to ensure a solid stance. One of the problems observed with stability was high latency in tracking the reference animation (see Figure 3.4.1). In order to quantitatively assess the latency, a representative joint was chosen (the right knee joint) and a correlation measure was calculated between the trajectory obtained in the simulation and the

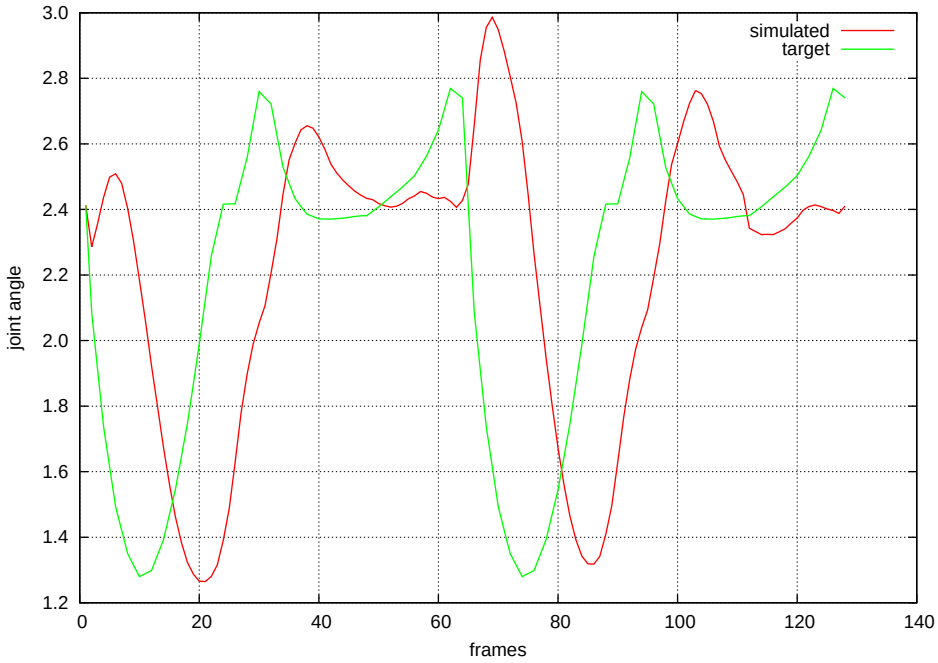

Figure 3.4.1: Tracking Latency in Right Knee Joint

reference animation offset by n frames, with n varying from 1 to 8. The correlation measure was based on the squared error between the target and actual joint angle values summed over 4 locomotion cycles. The correlation measure was plotted for several different joints and in each case a quadratic curve fit was found to describe the data well (see Figure 3.4.2). The latency could then be estimated by calculating the number of frames corresponding to the minimum of the quadratic. Code was added to the implementation to output this latency measure and the R^2 coefficient of the curve fit as a check that a quadratic curve gave a good fit to the data.

One factor thought to contribute to the latency in joint tracking was that the physics node used as the basis for the implementation did not set target angular velocities for joints, only target orientations. Code was added to supply target angular velocities to the joint drive calculated from subsequent frames of the input animation, and this improved the latency from 6.2 frames to 2.4 with the joint strength and damping settings used.

The effect of the joint drive strength and damping parameters on latency was also investigated by running the simulation repeatedly with different parameter values. The strength and damping parameters are scaled in the animation engine relative to values which have previously been tuned to give lifelike behaviour for other physics nodes, with both multipliers expected to have values close to 1.0 for a natural appearance. The latency values for a range of strength and damping multipliers are shown in Figure 3.4.3. The error bars are based on running the simulation 10 times for each condition. Note however that for small estimated values of latency, since the minimum in the quadratic fit is very close to one end of the data interval it

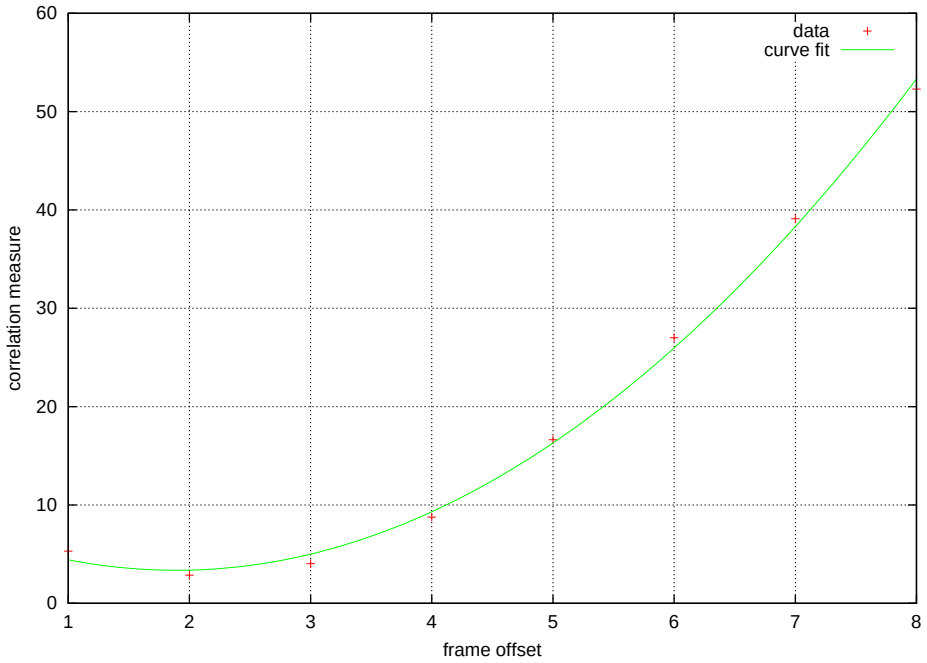

Figure 3.4.2: Curve Fit to Quantitatively Estimate Tracking Latency

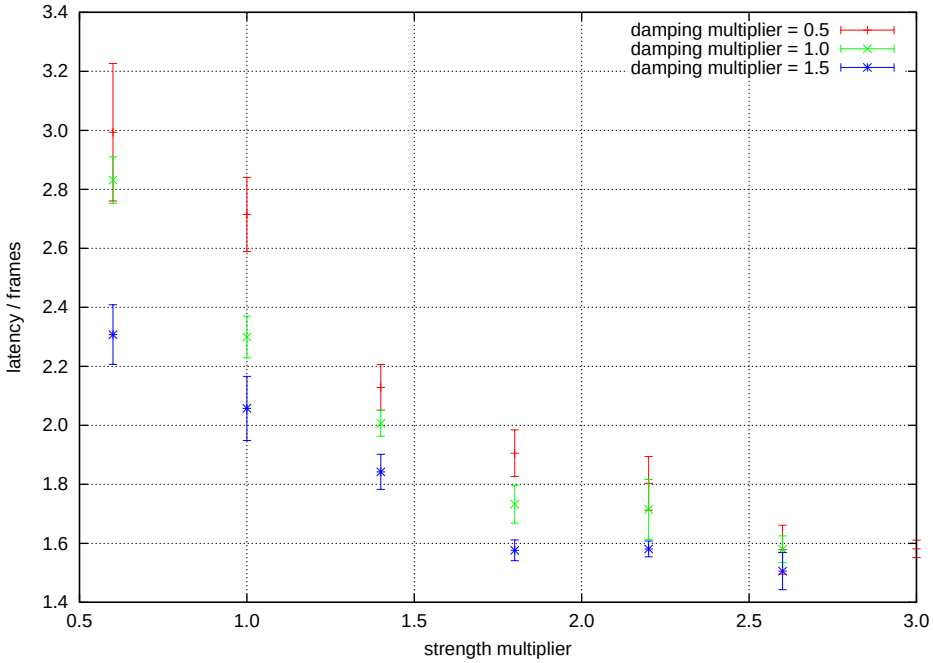

Figure 3.4.3: Effect on Latency of Joint Strength and Damping Parameters

may be quite weakly determined, and the small latency values may be less reliable than the error bars suggest. The results show that latency decreases with both increasing joint strength and increasing joint damping. However, large values of these parameters have other deleterious effects, with motion appearing stiff and unnatural, and for very large values render tracking completely unstable. Given the limited benefit to latency and deterioration in stability and naturalness for larger values, it appears that values close to the morpheme default values (e.g. multipliers in the range 1.0 — 1.5) are optimal.

3.5 Feedback Error Learning

The original SIMBICON paper (Yin et al. 2007) uses feedback error learning (FEL) to learn feed-forward torques, with the aim of reducing motion stiffness by allowing lower gain values to be used in the feedback PD control. In the FEL approach, feed-forward control is adapted over a series of locomotion cycles, by applying the current estimate of the feed-forward torques to the character and calculating a feedback correction to deal with the remaining deviation. The feed-forward estimates τ_{ff} are then updated by making a linear combination of the existing estimate with a small contribution from the feedback component τ_{fb}

$$\tau'_{ff} = (1 - \alpha)\tau_{ff} + \alpha(\tau_{ff} + \tau_{fb}) \tag{3.4}$$

where α is a parameter which affects the learning rate. The feed-forward estimates should converge over a series of locomotion cycles.

A framework to support the FEL approach was included in the morpheme implementation. As it was desirable to have the option of experimenting with different update rates for the physics simulation, the FEL framework was developed in a frame rate independent fashion, by dividing the locomotion cycle into a number of intervals of equal duration (64 intervals were used), with feed-forward estimates constant throughout each interval. For each frame of simulation, the applied feed-forward torque was interpolated by forming a weighted combination of the intervals which overlapped the duration of the frame, and updates to feed-forward torques were made in a similar weighted fashion.

The FEL method needs a stable feedback controller to work from, but the stability of the initial morpheme implementation was poor. Therefore, an attempt was made to produce a more stable basis for the FEL process by overriding the state of the physics model after each frame, with joint orientations and angular velocities determined

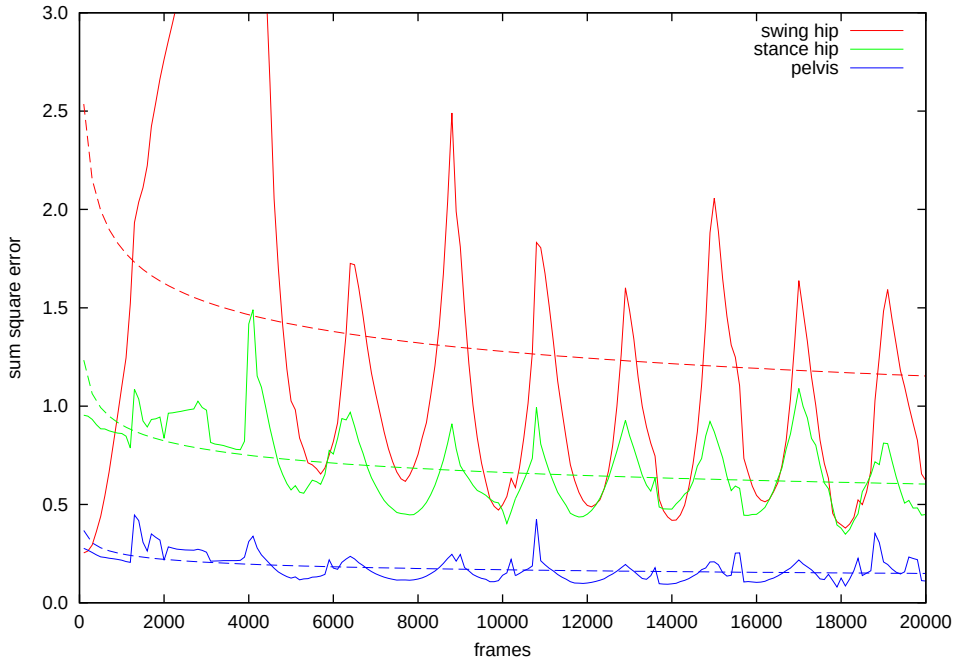

Figure 3.5.1: Feedback Error Learning Tracking Errors

from the reference animation. The tracking error was calculated by summing the squared error in joint orientation over one cycle of locomotion, and is shown over a large number of cycles of FEL with $\alpha = 0.05$ in Figure 3.5.1. Of obvious note are large spikes in the error with a fairly regular period of around 2000 frames. This period is much larger than the duration of the input animation and so does not correspond to any feature of the reference motion, and the origin of the spikes is not understood. There is some evidence of a downward trend in the tracking errors as a result of the FEL, but it is difficult to separate from the noise associated with the spikes in error, and the improvement, if any, is slow. Yin et al. (2007) do not describe how many cycles of FEL were required for convergence in their implementation. It is likely that the stability of the base controller is still too low to apply FEL effectively.

3.6 Angular Momentum Regulation

Another area which was explored in the initial experimentation was the incorporation of information about the angular momentum of the character about its COM into the control strategy. Several papers (Macchietto et al. 2009, Lee & Goswami 2010, Herr & Popovic 2008) emphasize that regulating angular momentum is an important aspect of control. Two sets of control variables can be considered equivalent — either the rate of change of linear and angular momentum, or the ground reaction force and centre of pressure. Thus it may be possible that the problems with ground contacts observed in the initial controller implementation could be addressed using angular momentum information.

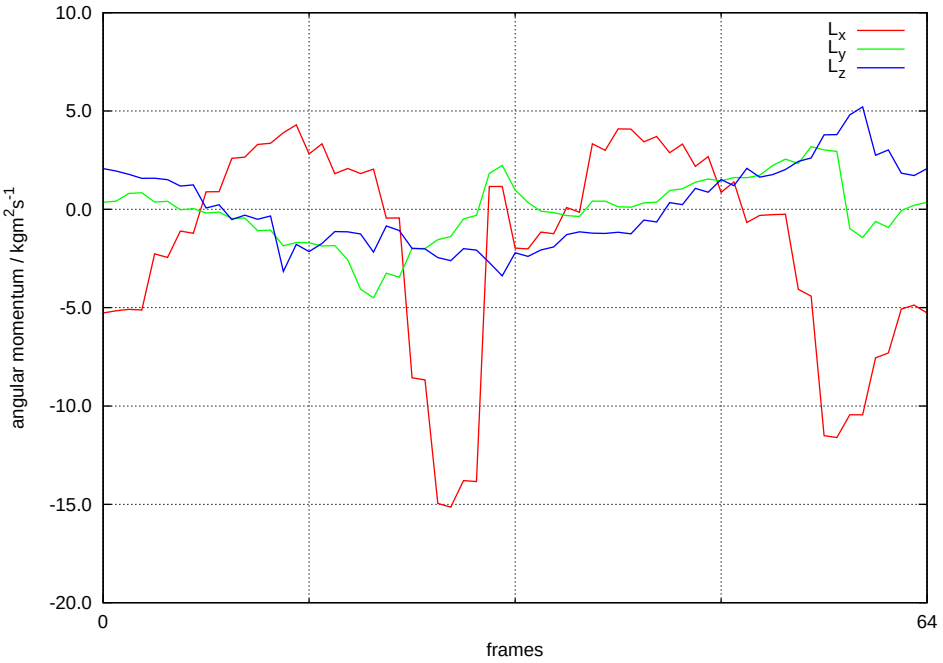

Figure 3.6.1: Reference Animation Whole Body Angular Momentum

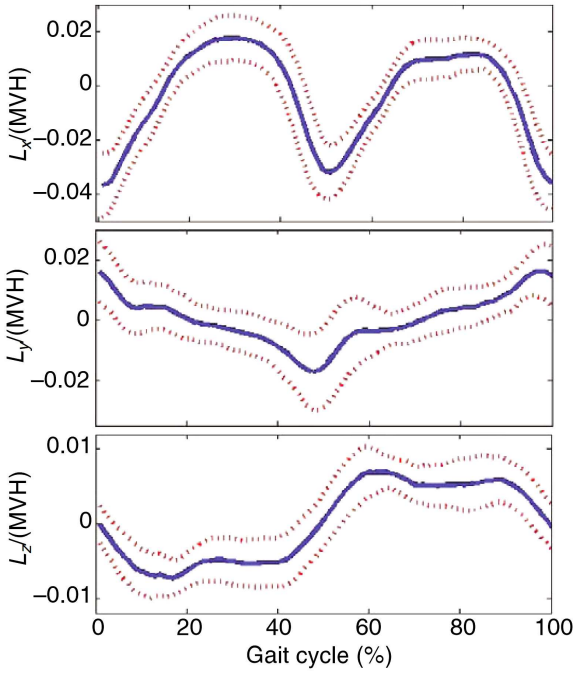

Figure 3.6.2: Measured Whole Body Angular Momentum (Reproduced from Herr & Popovic (2008))

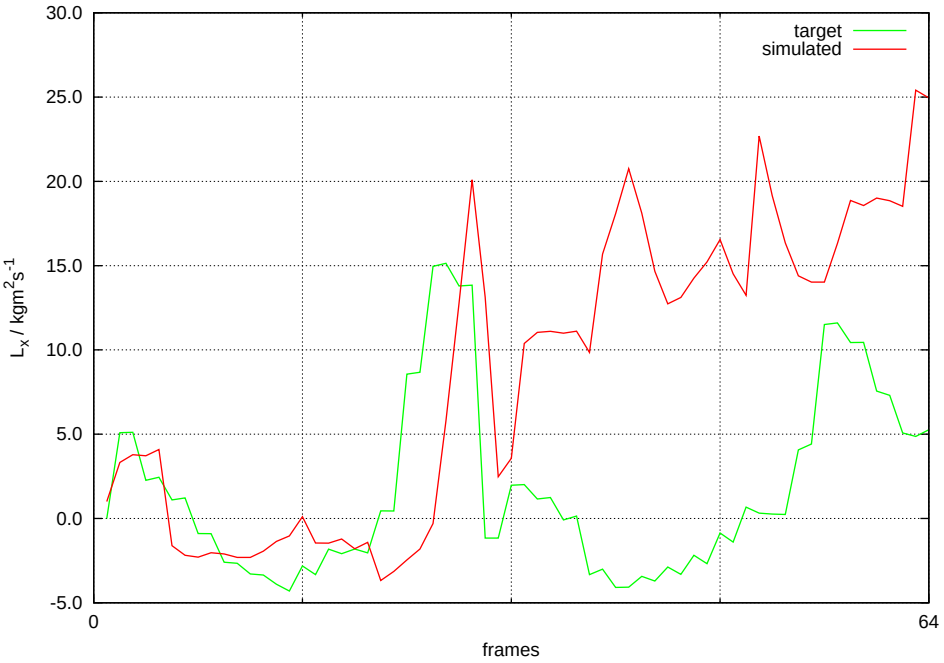

Figure 3.6.3: Whole Body Angular Momentum Tracking

It may be worthwhile to explore strategies similar to the approach described by Lee & Goswami (2010), in which optimal values for the GRF and COP are calculated, which are used to determine target joint accelerations. The joint torques required to achieve the desired accelerations are then calculated using inverse dynamics. One option which was considered here was to include the target and simulation angular momentum, and the GRF and COP from previous frames as inputs for the PD controller. By modifying the foot torques, it should be possible to manipulate the COP and thus improve balance. For example, in some instances in the initial implementation too much energy is being lost due to the impact of the leading foot with the ground, so that the character tends to tip over backwards. With feedback into the PD control, this would appear as a shortfall in the angular momentum about the COM compared to its target value, which could be compensated for by adjusting the torque on the trailing foot to push the COP backwards.

As the first step in implementing this idea, code was added to determine target values for the angular momentum about the character's COM based on the input animation, and to calculate the angular momentum in the physics simulation at each frame. As a test that the values returned were sensible, the output target values (Figure 3.6.1) were compared with whole body angular momenta reported by Herr & Popovic (2008), which were estimated from experimental measurements on 10 subjects (Figure 3.6.2). Note that the morpheme implementation has a y-up coordinate scheme, while Herr & Popovic (2008) use z-up. The form of the calculated target values is consistent with the experimental data.

Figure 3.6.3 shows the target and simulation angular momenta in the sagittal plane.

The tracking is reasonably faithful for the first half of the locomotion cycle, but is subsequently poor. It is thought that this poor angular momentum tracking is contributing significantly to the instability of the controller. However, as incorporating angular momentum information into the control would add more parameters, and manual tuning of the controller was already difficult and time consuming, it was decided to implement a more sophisticated automated parameter optimization framework before further work on angular momentum regulation.

3.7 World Space Control for Feet

A further source of instability in the initial controller implementation was the orientation of the feet. Poor orientation prior to heel strike often led to tripping, and poor orientation after foot placement interfered with achieving a robust stance. An attempt was made to improve the alignment by including the feet in the set of joints controlled in world space. Additional terms were also included in the balance feedback mechanism to encourage better foot placement leading up to heel strike. These measures were somewhat successful in increasing the number of steps the controller could take before becoming unstable, as can be seen in the video accompanying this section, but very large PD gains were required resulting in very stiff movement and jerky impacts following ground contacts.

3.8 Gravity Compensation

The large values for the gain parameters were in part necessary because the PD controller was correcting, in a feedback fashion, deviations from desired joint orientations caused by gravity. Explicit gravity compensation could be used to ameliorate this problem, and is included in NaturalMotion's behaviour system, euphoria. Furthermore, adaptation of the brain's internal model of dynamics under force fields (Shadmehr & Mussa-Ivaldi 1994, Lackner & Dizio 1994) suggests that gravity compensation is consistent with biological control, and could be considered as a form of feed-forward control.

The following method was implemented in morpheme to determine gravity compensation torques. For each joint, the mass and COM of the subtree of the physics rig associated with that joint is calculated, and the gravitational torque about the joint position evaluated. A compensating torque of equal magnitude to the gravitational torque is then applied to the physics part which is the immediate child of the joint, and an equal and opposite torque applied to the parent part of the joint to ensure

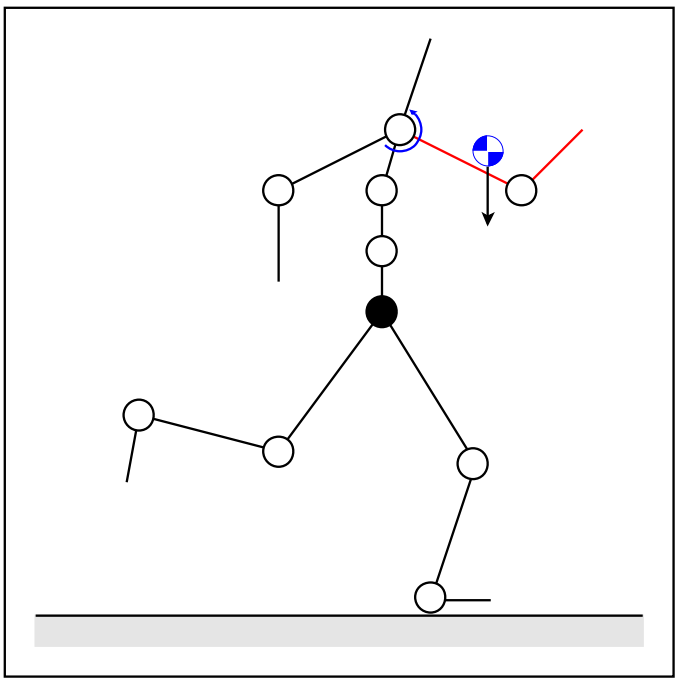

Figure 3.8.1: Gravity Compensation Torques

torques are balanced. The method is illustrated in Figure 3.8.1 for the shoulder joint, with its subtree highlighted in red. This procedure is iterated from the leaf joints inwards until the pelvis (shown in black in the diagram), which is considered as the root of the upper body, is reached. If one of the legs is not in contact with the ground, the joints in that leg are also included iteratively from the foot inwards until the pelvis is reached, and then gravity compensation continues down the other leg to the ground. If both feet are supported, the torques from the upper body must be distributed between the two supports, by calculating torques proportionally to the magnitude of the GRF at each foot.

With gravity compensation, it was possible to greatly reduce the PD controller gains, resulting in much smoother, but still poor, motion. The resulting motion is illustrated in the video accompanying this section.

3.9 Summary

Initial experimentation with large time step simulation was conducted based on the well-known SIMBICON framework. The original SIMBICON operates at 2000 simulation time steps per second and becomes unstable below about 750 time steps per second. Implementing the basic SIMBICON method at 60 time steps per second produced a controller stable for no more than one walking step, with problems due to tripping, dealing with sudden impulses from ground contacts, and providing sufficient forward momentum to complete steps. Foot constraints allowed many stable

steps to be performed, but resulted in very unnatural behaviour. Tracking latency was a major source of control difficulty, and although it was possible to improve latency by including angular velocity tracking and tuning the joint drive parameters, tracking delays remained which were significant compared to the duration of some phases of the reference motion. Feedback error learning also appeared to improve tracking slightly, but the base controller was too unstable to make FEL truly effective. World space foot control and gravity compensation also appeared to result in modest improvements, but motion quality still remained poor.

Overall, despite numerous attempts to improve the motion quality, it appears that it would be very difficult to implement a SIMBICON type controller effectively with large time steps, and that more sophisticated control approaches are required.

Chapter 4

Trajectory Optimization

4.1 Introduction

As described in Chapter 3, SIMBICON type controllers were found to be very problematic with large time steps, and it appears that more sophisticated control approaches may be necessary to achieve stable control under such conditions. One issue which may cause such methods to fail is attempting to track reference motions which lack physical validity. This chapter describes the use of trajectory optimization techniques to improve the physical validity of reference motions, in order to increase their suitability for tracking with physics-based controllers.

In order to achieve lifelike motion with physics-based character control, it is often desirable to track a reference such as a motion capture sequence. However, due to noise in the capture process and artefacts introduced by retargeting the actor's motion to a character model, the physical validity of the reference may be degraded, causing the tracking to fail. In particular, the position and timing of contacts with the environment may be incorrect. Therefore, it is useful to be able to process motion sequences, and their contact configurations, to improve physical plausibility whilst retaining the subtle details which contribute to realism.

This problem can be tackled with trajectory optimization, a process used to determine control sequences for a system which are optimal in terms of an objective function, subject to constraints. Here, the objective function might reward closeness to the original reference motion, with the constraints enforcing physical plausibility. There are two general families of approaches to trajectory optimization, shooting methods and direct methods. In shooting methods, the space of control policies is searched, with the state trajectory corresponding to a given control sequence determined by forward simulation subject to the system dynamics. In direct methods,

optimization takes place over both the control and state trajectories simultaneously, with constraints ensuring consistent dynamics.

Shooting methods can handle modifications to contact configurations, but are very sensitive to changes near the beginning of the control interval, as their influence propagates over the whole interval, making such problems difficult to solve.

Contacts are difficult to include as variables in direct methods as they are hard to formulate in differentiable form, and as a result much previous work with direct methods uses fixed contact schedules. Posa et al. (2014) do incorporate contacts into direct trajectory optimization, but at the expense of introducing complementarity constraints. A complementarity constraint enforces that the product of a pair of variables is zero, and these constraints make problems difficult to solve as the feasible region of the search space is poorly connected. Posa et al. (2014) use a constraint relaxation method, allowing the product of the complementary variables to take a small non-zero value, which is then driven to zero as optimization progresses. The approach is demonstrated for problems with complex contacts, although only planar examples are considered. In general, constraint relaxation methods can encounter difficulties when the relaxation parameter gets very close to zero.

This chapter extends the direct method of Posa et al. (2014) to 3D problems with contacts using an exact penalty method, in which a specially formulated term is added to the objective function which leads to the complementarity constraints being satisfied. The results suggest that the exact penalty method offers significantly greater robustness and performance compared to the constraint relaxation approach. The performance also appears strong compared to previous work incorporating contacts using shooting methods. As an auxiliary contribution, it is demonstrated that trajectory optimization problems can be effectively tackled using an interior point solver, as well as previously used sequential quadratic programming methods. This work also uses an open-source solver, allowing greater applicability than previous work relying on proprietary software.

4.2 Related Work

This section previous work on trajectory optimization, which is also described in more detail in Section 2.5.1.

Trajectory optimization (also described as spacetime optimization or spacetime constraints), as introduced by Witkin & Kass (1988), is the idea of manipulating motions over a whole interval by treating the series of poses as variables in a constrained

optimization problem. Physical consistency may be enforced by constraints, and desirable motion characteristics encouraged through an objective function. Full characters may have dozens of DOF for each time step in the interval, leading to costly optimization problems involving thousands of optimization variables overall, and much subsequent work includes measures to improve performance. An additional difficulty with trajectory optimization is that it is very susceptible to problems with local minima. Many of the papers discussed below also address that issue, but it is not the focus here.

Methods to improve performance include splitting motion into a series of independent subproblems (Cohen 1992), using more compact representations such as B-splines (Cohen (1992) and many subsequent papers) or wavelets (Liu et al. 1994), allowing the precise timing of keyframes to vary (Liu & Cohen 1995), and using simplified dynamics (Liu & Popović 2002, Abe et al. 2006) or low-dimensional character models (Popović & Witkin 1999, Safonova et al. 2004). However, these approaches are not guaranteed to produce results which are physically consistent between fixed time steps, which is desirable for tracking with physics-based controllers as required here.

Efficient solution of constrained optimization problems usually requires derivatives to be calculated, and the timing of foot contacts is difficult to express in formulations which are differentiable. This means that a prescribed contact schedule is often used during optimization, and none of the methods discussed above allow contact timings to be changed during optimization.

Variable contact timings can be achieved by performing trajectory optimization as an inner loop, and adjusting contacts in an outer loop (Wampler & Popović 2009), but this approach is very expensive computationally.

Modifiable contacts can be included in more tractable ways by using a continuous approximation to the discontinuous contact dynamics (Todorov 2011) or using soft constraints (Mordatch et al. 2012, 2013). Soft constraints are related to the exact penalty method presented in Section 4.4, with each approach replacing constraints with penalty terms in the objective function. However, the contact method described by Mordatch et al. (2012) is not guaranteed to completely eliminate contact errors. The method presented here allows the body of work on exact penalty methods to be drawn on, including convergence results, understanding which formulations of penalty functions might be expected to be driven to zero, and strategies for selecting weighting for penalty terms. This work therefore has value in elucidating the soft constraint approach and its connections with other fields. Furthermore, Mordatch et al. (2012) use objective terms to enforce physical consistency, allowing small physics violations to remain in the final result (their aim is to generate visually

plausible motions rather than concentrating on strict physical validity). In the method presented here, the aim is to produce motions highly suitable for simulated tracking, and physics violations, including contact violations, are reduced to zero (within the tolerance of the solver).

Posa et al. (2014) incorporate contacts as optimization variables by using complementarity constraints, which are awkward to solve as they make the search space poorly connected. They use a constraint relaxation method to deal with complementarity constraints, and demonstrate examples with complex contacts, although only in 2D. Solutions take up to an hour to obtain for a planar 13 DOF character.

This chapter focuses on direct trajectory optimization, but some other approaches are briefly discussed here. Muico et al. (2009) use an optimal control solver as a preprocessing step to make motion capture data more suitable for tracking. It is not clear whether the method used allows contacts to be optimized. Several hours of processing time are required for a character with 29 DOF. Al Borno et al. (2013) use a shooting method with trajectories synthesized in 0.5s windows using covariance matrix adaptation, requiring about 20 minutes per window with a 41 DOF character.

4.3 Overview of Trajectory Optimization

A general nonlinear optimization problem involves minimizing an objective function f subject to inequality and equality constraints g and h.

$$\begin{aligned} \min_{\mathbf{x}} & f(\mathbf{x}) \\ s.t. & g(\mathbf{x}) \geq 0 \\ & h(\mathbf{x}) = 0 \end{aligned} \tag{4.1}$$

where $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is a vector of unknowns to be solved for. Trajectory optimization casts the problem of manipulating motion data in this form, with the motion represented by $\mathbf{x}$ and physical consistency being enforced by the constraints. The objective function can then be used to encourage desirable characteristics in the output motion, such as energy efficiency or closeness to a reference trajectory.

4.4 Contact Formulation and Complementarity Constraints

Following the method introduced by Posa et al. (2014), contacts were incorporated as variables in the optimization problem. In extending the method to 3D, a friction cone formulation was initially used, but this made it awkward to constrain frictional forces to oppose motion in sliding. It also introduced severe nonlinearities, which often made the solver fail to converge. Using a friction pyramid formulation, as described by Stewart & Trinkle (1996), the solver behaved much more robustly. A four-sided pyramid was used here. A value of 1 was used for the coefficient of friction. This is somewhat higher than a realistic value, and so may make it easier to avoid slipping and allow exploitation of larger tangential friction forces, but this value was chosen for consistency with other NaturalMotion simulations, and it would be straightforward to explore other values in future.

To avoid specifying many redundant constraints, only points which were expected to make contact with the ground were included in the problem formulation. For all the examples considered here, this included four points on each foot. Other points could be added if necessary to deal with motions where other body parts such as hands or knees might contact the ground. No mechanism was included to deal with self-contacts.

The choice of contact formulation introduces complementarity constraints, i.e. constraints of the type

$$\begin{aligned} x_i &\geq 0 \\ g_i(\mathbf{x}) &\geq 0 \\ x_i g_i(\mathbf{x}) &= 0 \end{aligned} \tag{4.2}$$

This set of constraints may also be written using the more compact notation

$$0 \leq x_i \perp g_i(\mathbf{x}) \geq 0 \tag{4.3}$$

For a basic model of contact forces, the first inequality may specify that the separation between bodies is never negative (i.e. penetration does not occur), the second may specify that reaction forces may only be repulsive, and the final equality may specify that reaction forces can only apply when bodies are in contact.

Friction can also be incorporated into this framework as follows

$$\begin{aligned} 0 \leq y \perp c_n \geq 0 \\ 0 \leq \lambda \mathbf{e} + \mathbf{v} \perp \boldsymbol{\beta} \geq 0 \\ 0 \leq \mu c_n - \mathbf{e}^T \boldsymbol{\beta} \perp \lambda \geq 0 \end{aligned} \tag{4.4}$$

where y is the contact separation, c_n the normal force, $\mathbf{e}$ are vectors defining the friction directions (the friction polygon corresponds to the convex hull of $\mathbf{e}$), $\mathbf{v}$ are the components of the relative velocity of the bodies parallel to the friction directions, λ is a term related to the relative velocity, and μ is the coefficient of friction. The first complementarity constraint is similar to the basic model described above. The second and third constraints enforce that friction acts in a direction to prevent sliding and that the tangential forces satisfy the Coulomb friction model. See Stewart & Trinkle (1996) for full details.

Complementarity constraints are difficult for solvers as the feasible region is poorly connected, with discontinuous jumps between points with non-zero x_i and points with non-zero $g_i(x)$. Two common ways to deal with this issue are constraint relaxation, as used by Posa et al. (2014), and penalty methods.

In the constraint relaxation approach, the constraint is replaced with

$$\begin{aligned} x_i \geq 0 \\ g_i(\mathbf{x}) \geq 0 \\ x_i g_i(\mathbf{x}) \leq \alpha \end{aligned} \tag{4.5}$$

for a small value of α, which is reduced towards zero as the solution progresses. Even with constraint relaxation, difficulties can still be encountered with convergence when α gets close to zero. During experiments, convergence issues were found at α values comparable to the desirable complementarity tolerance, and so potentially interfering with reaching a solution.

Other methods are based on penalty functions. In these approaches, a modified problem is considered, in which the constraint is removed altogether and replaced with an additional weighted term in the objective function.

Fukushima et al. (1998) consider a sequential quadratic programming (SQP) ap-

proach using a differentiable penalty term with a smoothing parameter to handle complementarity constraints. Rather than solve a series of problems, they describe a method to update the penalty weight and smoothing parameter after each SQP iteration.

Many solvers include an elastic mode, in which constraints are relaxed, and a penalty term is added to the objective function. Anitescu (2005) shows that the elastic mode can be used effectively for problems with complementarity constraints.

It can be shown that, for certain formulations of penalty function and sufficiently large values of the penalty weight, solutions of the original problem are also solutions of the modified problem (Benson et al. 2006, Wright & Nocedal 1999, Ch. 15, Ch. 17.3). A formulation with this property is referred to as an exact penalty function. Choosing a suitable weight value in advance can be difficult, but it is possible to start with a moderate value and increase it in successive steps until the penalty term becomes sufficiently small.

Benson et al. (2006) describe two formulations of exact penalty functions for complementarity problems. In the ℓ_1 or summation formulation, a term of the form $\rho \sum x_i g_i(x)$ is added to the objective function, where ρ is the penalty weight. In the ℓ_∞ or pairwise formulation, the penalty term is of the form $\rho \max_i \{x_i g_i(x)\}$.

The summation formulation described by Benson et al. (2006) was originally used for this implementation, but this led to a problem. Within the solver tolerance, it was possible for each $x_i g_i(x)$ in the sum to have a small negative value. The solver would succeed in driving the summed penalty term to zero, but since there might be thousands of complementarity constraints in the problem, a large number of small negative values could allow a few of the individual terms in the sum to have large positive values and hence significantly violate their constraints. Changing to a pairwise formulation eliminated this problem, and with this amendment the exact penalty method was found to be more robust than the constraint relaxation implementation for solving complementarity problems. The pairwise formulation replaces the objective function with

$$\begin{aligned} & f(\mathbf{x}) + \rho\zeta \\ & x_i g_i(\mathbf{x}) \leq \zeta \end{aligned} \tag{4.6}$$

where f is the original objective function, and ρ is the penalty weight. The inequality applies for each complementarity constraint, so ζ gives the maximum complemen-

tarity violation.

4.5 Optimization Algorithm

The two leading methods for solving large nonlinear optimization problems are sequential quadratic programming (SQP) and interior point methods (Wright & Nocedal 1999). Both methods essentially use Newton steps to iteratively improve solutions, but they differ in how constraints are handled. Typically in SQP, a list of which inequality constraints are at their limits (the active set) is iteratively updated based on Lagrange multiplier estimates. Interior point methods use logarithmic barrier terms to prevent iterates getting too close to inequality bounds until the optimization has progressed sufficiently. Based on available results in the literature, both methods appear to be fairly close in terms of robustness and performance. For example, Betts & Gablonsky (2002) suggest that SQP methods have a slight advantage. On the other hand, results for a large number of standard test problems reported by Benson et al. (2003) suggest that the interior point methods LOQO and KNITRO achieve better performance and robustness than the SQP code SNOPT. However, these comparisons are not particularly recent and updates have been made to both methods. Unfortunately, no more recently published comparisons could be found.

The majority of existing work on trajectory optimization uses the SQP approach, in particular the proprietary software SNOPT. However, in this work it was desirable to develop code which could be used commercially without licensing costs. The free open-source SQP solver HQP (Franke & Arnold 1999) was considered, but it was not used as it does not appear to have been maintained for some time, and the documentation is limited. No other suitable free SQP solvers could be found.

The free, open-source interior point code IPOPT (Wächter & Biegler 2006) was used as it is high quality, well tested and actively maintained, with good documentation.

IPOPT requires the first derivative of constraints and first and second derivatives of the problem Lagrangian. Symbolic differentiation was used in order to take advantage of problem sparsity. ADOL-C (Walther & Griewank 2012) was initially used and performed well for the majority of the function evaluations, but determining the Lagrangian Hessian was very slow, taking the overwhelming majority of the evaluation time and dominating the overall solver time. The slow performance may be associated with some detail of the problem formulation, but was not explored in sufficient detail to determine a specific reason. Switching to the alternative symbolic code CasADi (Andersson 2013) eliminated this performance issue, reducing function evaluation to an insignificant proportion of the total solution time.

MUMPS (Amestoy et al. 2001) was used as the linear solver within IPOPT. Results suggest that other solvers may be 2 or 3 times faster (Gould et al. 2007) but MUMPS was used as it is public domain. The single threaded version of MUMPS was used, as the documentation suggests that the overhead of parallelization is likely to eliminate any benefit of multithreading for the size of problems encountered here. METIS (Karypis & Kumar 1998) was used to improve matrix ordering. Using OpenBLAS rather than the default BLAS implementation included with MUMPS improved solution time by about 40%.

The implementations of both approaches to complementarity constraints involved successive runs of the solver, with updated values of either the relaxation parameter α or the penalty weight ρ. Each subsequent run was initialized with the optimized variable and Lagrange multiplier estimates from the previous run. IPOPT pushes initial values away from variable boundaries to ensure feasibility, but the default value for the size of the boundary push (0.01) was quite severe, and would cause some progress towards the solution from the previous run to be lost. Therefore the boundary push value was reduced to 10^{-9}.

IPOPT offers two strategies for reducing the interior point barrier term as the solver progresses, monotone and adaptive. Generally the adaptive method reduced the required number of iterations, but occasionally it would affect stability. Using the monotone method for the first solver run (where the initialization might be relatively far from the optimal solution), and the adaptive method for subsequent runs appeared to be a good compromise.

4.6 Implementation Details

4.6.1 Character Model

A 20-link character model was used with identical link arrangement, dimensions and inertia to the default morpheme physics rig used in Chapter 3. The joint limits from that physics model were not replicated for this work, and all the joints were treated as ball joints so, counting the position and orientation of the centre of mass, the model had a total of 63 degrees of freedom. It would be possible to include joint limits in the trajectory optimization model, but that would lead to many additional constraints, and should not be necessary as the poses used for initialization can guide the optimization to configurations within a suitable range of motion.

4.6.2 Dynamics Formulation

For simplicity of implementation, a maximal co-ordinate representation was used to describe the character's dynamics. Although reduced co-ordinate representations such as Featherstone's algorithm could substantially reduce the number of variables, the resulting optimization problem would also be much denser, and therefore would not be expected to change performance much while adding significant complexity.

Previous work often uses splines to reduce problem size. Splines were not used here, for simplicity, because selecting control points would be problematic with the flexible contact scheduling, and because the main motivation was to produce physically processed reference motions, for which densely sampled time steps were required.

The character's state at each time step was described by the position and orientation of each link, the force and torque at each joint, and the details of the contact forces. The position of each link was specified by the co-ordinates of its centre of mass to make rotational dynamics easier. Quaternions were used to represent orientations, since they are relatively compact (4 parameters for 3 DOF), but reasonably simple algebraically. Quaternion normalization was maintained by solver constraints. Constraints were also used to ensure that for each link, the quaternions for successive time steps were in the same hemisphere, which simplified the algebra for calculating angular velocities and accelerations.

Joint alignment was enforced by constraining the position of each link relative to its parent at each time step.

Physical validity was ensured by constraints equating the finite difference estimate of the linear acceleration of each link at each time step to the sum of forces acting on it, with a similar constraint relating the rate of change of angular momentum and total torque.

$$\begin{aligned} \mathbf{p}_{t+1} - 2\mathbf{p}_t + \mathbf{p}_{t-1} &= \mathbf{g}\Delta t^2 + \frac{\Delta t^2}{m_l}\left(\sum_j \mathbf{f}_{tj} + \sum_c \mathbf{f}_{tc}\right) \\ \frac{\mathbf{I}_{t+1}\omega_{t+1} - \mathbf{I}_t\omega_t}{\Delta t} &= \sum_j \mathbf{r}_{tj} \times \mathbf{f}_{tj} + \sum_c \mathbf{r}_{tc} \times \mathbf{f}_{tc} \end{aligned} \tag{4.7}$$

where $\mathbf{p}_t$, $\mathbf{I}_t$ and ω_t are the COM position, moment of inertia and angular velocity at time t, $\mathbf{g}$ is the acceleration due to gravity, m_l is the link mass, and $\mathbf{f}_{tj}$ and $\mathbf{f}_{tc}$ are joint and contact forces, applied at positions $\mathbf{r}$ relative to the link COM.

If the output motion was not required to be cyclic, the constraints would extend over time steps $t = 2 \ldots T - 1$. For cyclic motions, the constraints would apply to all of the time steps, with quantities spanning the end of the interval transformed to take the rotation and translation between successive cycles into account.

As described in Section 4.6.1, the problem formulation did not include any model of the physiological limits on the range of motion of the joints. Joint orientations at each time step were constrained to be similar to those in the corresponding time step of the initialization (within a rotation of about 10°), but the purpose of this was to reduce the size of the search space for the solver and avoid getting stuck in blind alleys, rather than limit joint configurations. Similarly, link positions were constrained to within a cube of half-side 20cm centred on the initialization pose. No upper limits on joint forces or torques were specified.

4.6.3 Objective Function

The objective function for the optimization comprised a weighted sum of generally quadratic terms designed to encourage desirable features in the output motion. The first part of this section describes the main features of the objective function used for improving the physical validity of a reference motion. Some amendments to the objective function for motion editing tasks are described at the end of this section.

For each link, deviations of the link centre of mass position, $\mathbf{p}$, from target poses were penalized by the term

$$\sum_{t=1}^{T} \sum_{l=1}^{n_l} m_l \left\| \mathbf{p}_{tl} - \bar{\mathbf{p}}_{tl} \right\|^2 \tag{4.8}$$

where the sum extends over T time steps and n_l links and barred terms represent target values from the reference motion. The terms were scaled by link mass m_l to give more influence to parts such as the hips and torso.

A separate term penalized deviations from the target position for the character's overall centre of mass.

$$\sum_{t=1}^{T} \left\| \sum_{l=1}^{n_l} m_l \left(\mathbf{p}_{tl} - \bar{\mathbf{p}}_{tl} \right) \right\|^2 \tag{4.9}$$

The objective function also included a term penalizing discrepancy from target link orientations. For ease of implementation, the term was based on the scalar part

$S(\cdot)$ of the quaternion mapping between simulated and target orientations, $\mathbf{q}$ and $\bar{\mathbf{q}}$, which behaves as a quadratic function of the angular error for small discrepancies.

$$\sum_{l=1}^{n_l} m_l \left(1 - S\left(\mathbf{q}_{tl}\bar{\mathbf{q}}_{tl}^*\right)\right) \tag{4.10}$$

With the objective terms discussed so far, output motions were very jittery. Attempting to address this, a term based on minimizing the summed square of link linear accelerations was added, but this led to undesirable behaviour in which successive links in chains oscillated in alternating directions. Adding a term encouraging link angular velocities ω to be similar to those from the target motion was more successful.

$$\sum_{t=1}^{T-1} \left\|\omega_t - \bar{\omega}_t\right\|^2 \tag{4.11}$$

As a measure of energy expenditure, the sum of squared joint torques τ over the n_j joints was included to encourage more natural behaviour.

$$\sum_{t=1}^{T} \sum_{j=1}^{n_j} \left\|\tau_{tj}\right\|^2 \tag{4.12}$$

There was a tendency for contact forces and torques to be applied intermittently. To avoid this, terms were added to the objective function to encourage little variation between successive time steps.

$$\begin{aligned} &\sum_{t=1}^{T-1} \sum_{c=1}^{n_c} \left\|\mathbf{f}_{t+1c} - \mathbf{f}_{tc}\right\|^2 \\ &\sum_{t=1}^{T-1} \sum_{l=1}^{n_l} \left\|\tau_{t+1l} - \tau_{tl}\right\|^2 \end{aligned} \tag{4.13}$$

For physical processing of reference motions, the position and orientation objectives were calculated in world space. For motion editing (see Section 4.7.4), the position terms were expressed relative to the character COM, with a frame orientated using the COM velocity, and angular terms were formulated for each link relative to its

parent. To avoid unnatural torso orientations when editing slope angle, an additional objective function term was included based on the position of the character's COM projected onto the ground relative to the foot positions.

The weights were chosen to make the contributions of each of the terms to the overall objective function approximately equal. Some of the weights were tuned over successive runs if the resulting output appeared to place too little or too much emphasis on the associated motion characteristics. Relatively little tuning effort was required. Values for the weights are given in Table 4.6.1. These are the values before scaling was applied (see Section 4.6.4).

Objective Term	Unscaled Weight
Link Position	50
COM Position	10
Link Orientation	10
Angular Velocity	0.1
Sum Squared Torque	1E-5
Contact Variability	2E-5
Torque Variability	2E-6

Table 4.6.1: Values Used for Objective Function Weights

4.6.4 Problem Scaling

Nonlinear solvers can be sensitive to problem scaling. For example, progress may be very slow in directions where derivatives are similar in size to the solution tolerance. Therefore it is desirable for all the non-zero derivatives to be comparable in magnitude. Solvers may include scaling strategies, but it is difficult to automate scaling as derivatives vary with the current solution estimate and the derivatives at the initialization point may differ significantly from those near the solution. For this reason, it is often useful to perform manual scaling using knowledge of the problem.

The trajectory optimization problem was originally formulated using units of kilograms, metres and seconds, but this led to very poor problem scaling. For example, a linear acceleration constraint as formulated in Equation 4.7 might contain terms of typical size $\mathbf{g}\Delta t^2 = 9.81ms^{-2}(0.0167s)^2 = 2.7 \times 10^{-3}m$. On the other hand, a constraint ensuring that contact forces are not negative might have terms of typical size $100N = 100kgms^{-2}$, nearly 5 orders of magnitude larger.

The formulation was improved by using units corresponding to characteristic mass, length and time values. Typical values were estimated for the quantities involved in each kind of constraint and each term in the objective function, and a simple least squares problem was constructed to determine characteristic mass, length and

time values which minimized the variation in magnitude. Each scaled term s_i was written in terms of its unscaled equivalent u_i

$$s_i = {s_m}^{n_m^i} {s_l}^{n_l^i} {s_t}^{n_t^i} u_i$$
$$ln(s_i) = n_m^i ln(s_m) + n_l^i ln(s_l) + n_t^i ln(s_t) + ln(u_i) \tag{4.14}$$

where s_m, s_l, s_t are the scaling factors for mass, length and time, and n are the exponents of each dimension in the term. Then the scaling factors which bring the scaled quantities closest to 1 were found by minimizing

$$\sum_i (ln(s_i))^2 \tag{4.15}$$

Rounding the least squares estimates gave the values 8kg, 0.5m, and 0.025s. With these units, the example terms above scale to

$$(2.7 \times 10^{-3} m)(0.5m)^{-1} = 5.4 \times 10^{-3}$$
$$(100 kgms^{-2})(8kg)^{-1}(0.5m)^{-1}(0.025s)^2 = 1.6 \times 10^{-2}$$

which are much closer in magnitude.

4.7 Results

Testing was performed using a Core i7-3770 with single-threaded code compiled under Cygwin. A time step of 1/60 of a second was used for all the examples.

4.7.1 Scaling

Testing was performed with and without the problem scaling scheme described in Section 4.6.4. Table 4.7.1 shows the total time and number of iterations for a complete sequence of solver runs for a 64 time step cyclic walk motion, using the exact penalty method to handle complementarity constraints.

Scaling	Iterations	Time / h	Objective
no	5196	8.7	763.0
yes	1654	1.2	126.4

Table 4.7.1: Solver Performance With and Without Problem Scaling

4.7.2 Complementarity Approaches

The two methods for dealing with complementarity constraints were tested using the cyclic walk reference. Complete sequences of successive solver runs for the constraint relaxation approach and exact penalty method are shown in Table 4.7.2 and Table 4.7.3 respectively. The starting values of the relaxation parameter, $\alpha = 10^{-4}$, and the penalty weight, $\rho = 10^5$, were chosen based on experience.

α	Iterations	Time / h	Objective
1E-4	348	0.32	112.3
1E-5	509	0.50	123.1
1E-6	150	0.13	124.4
1E-7	633	0.67	restoration failed
1E-8	2589	2.51	124.9

Table 4.7.2: Solver Output for Successive Steps of the Constraint Relaxation Method

ρ	Iter	Time / h	Objective	Maximum Complementarity Violation
1E5	292	0.21	121.6	2.1E-04
1E6	115	0.08	126.4	7.4E-07
1E7	127	0.12	125.6	1.0E-20

Table 4.7.3: Solver Output for Successive Steps of the Exact Penalty Method

The two methods were further tested using cyclic jogging and running motions. The results are shown in Table 4.7.4. For time reasons, the relaxation method was terminated at a complementarity violation of 10^{-8}.

4.7.3 Cyclic Walk

Processing a reference motion to improve its physical validity was tested using a cyclic walking motion with a duration of 64 time steps. The reference motion was used both to supply target poses for the objective function and to initialize the solver. Using the exact penalty method, the solver required just under 25 minutes to reach an optimal solution which satisfied the physical constraints. As a measure of similarity between the optimized trajectory and the reference motion, Table 4.7.5 shows the typical deviations from target poses. Typical frames from the original and optimized motions are shown in Figure 4.7.1, and the motions are also compared

Method	Exact Penalty		Relaxation	
Motion	jog	run	jog	run
Frames	42	34	42	34
Total Iterations	509	759	1300	1198
Total Time / h	0.35	0.45	1.68	0.86
Final Objective	463.7	678.0	495.3	671.2
Complementarity	1E-20	1E-20	1E-8	1E-8

Table 4.7.4: Additional Comparison of Exact Penalty and Constraint Relaxation Methods

in the accompanying video. As desired, the optimized motion appears very similar to the reference, but corrects problems with physical validity, most noticeably in the height of the feet above the ground, as shown in Figure 4.7.2. Additionally, Figure 4.7.2 shows that the solver is able to modify contact timings to reach an optimal solution, which would not be possible without including contacts in the optimization variables. Of course, techniques such as inverse kinematics could be used to eliminate the problems with foot height but significant care would be required to avoid degrading the physical validity in other areas.

Quantity	Typical Deviation
Link Position	18mm
COM Position	1.1mm
Link Orientation	0.69°

Table 4.7.5: Typical Deviations from Target Poses for Cyclic Walk Motion

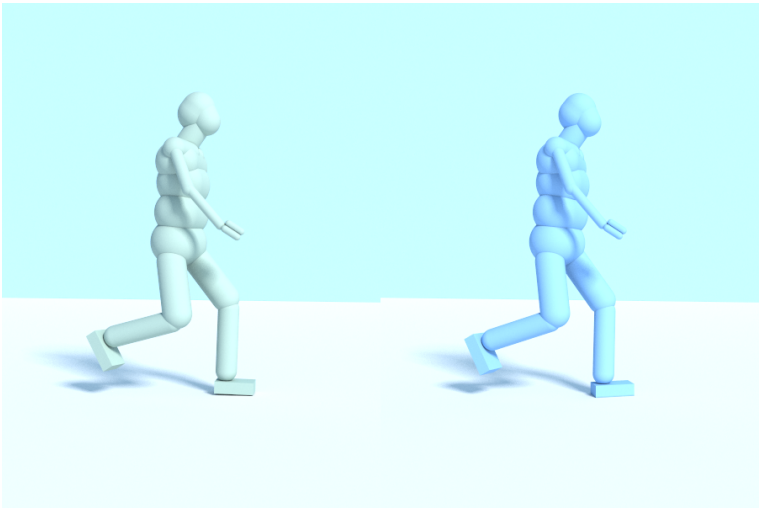

Figure 4.7.1: Typical Frame from Reference Motion and Corresponding Optimized Frame

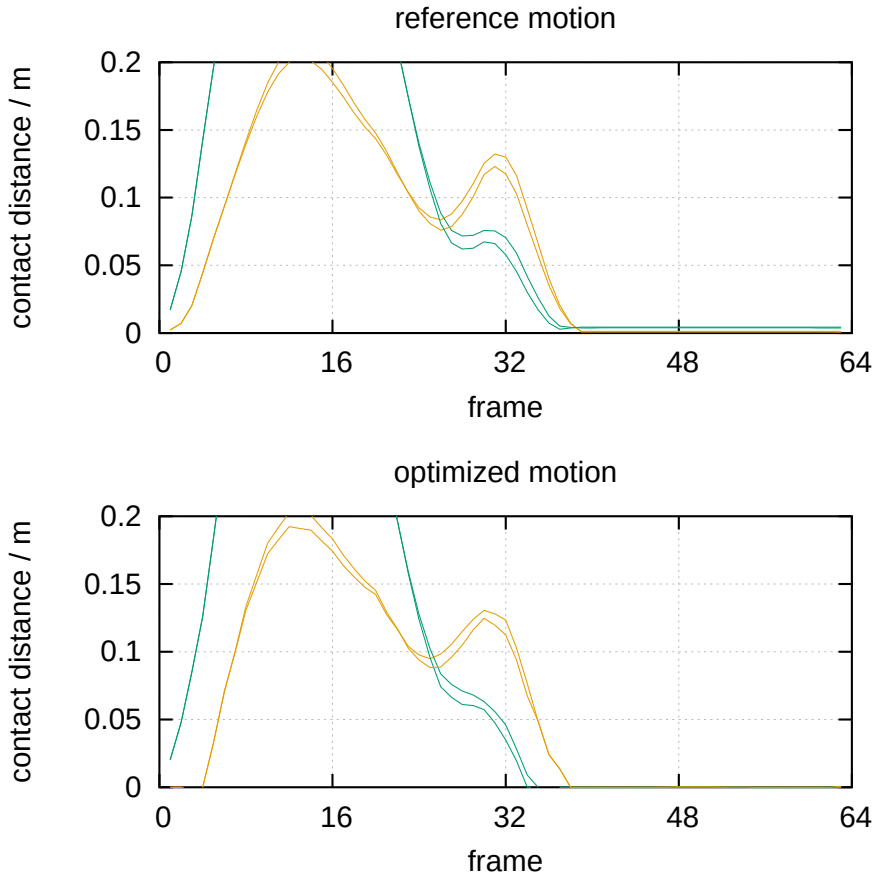

Figure 4.7.2: Contact Separation for Front (Orange) and Rear (Green) Corners of Right Foot Physics Body

4.7.4 Motion Editing and Transition Synthesis

Although the chief motivation for using trajectory optimization was improving the physical plausibility of references, investigations were also performed using the solver to alter aspects of the motion including slope angle ($-$20-20°), heading direction (0-150°), stride length (110-160cm) and step duration (34-64 frames), and to generate transitions between two animation segments.

Alterations of slope were constructed by changing the angle of the gravity vector. Other motion properties were edited by modifying the offset translation and rotation used in the cyclic constraints. Changes were made in small increments (5° for slopes, 10° for heading direction, 5cm for stride length, 3 frames for motion duration), with each new motion initialized using the previous result. Simple time warping was used to create initializations of appropriate duration when altering motion timing.

A transition from a forward jog to a banking jog, and a transition from an idle pose to a walking motion were generated. In each case, the two animation segments were used to provide target poses for the objective function during the initial and final portions of the interval, with no target poses specified for the duration of the

transition. The solver was initialized using a simple blend between the final pose of the first segment and the initial pose of the second segment, with a sigmoid function used to smoothly vary the blend weight during the transition.

As described in Section 4.4, the problem formulation did not include any measures to prevent character self-collision. For physically processing motion, this was not an issue as the target poses naturally avoid self-collision. However, for motion synthesis tasks, incomplete target pose information could lead to self-collision. To avoid this, ad hoc constraints were added to ensure a minimum separation between the feet, and between the hand and hip links.

Similarly, no joint limits were specified in the problem formulation, and synthesis tasks could result in unnatural configurations, in particular backwards bending of the knee. An objective function term was added based on the dot product of the direction of the perpendicular from the hip-ankle line to the knee joint, and the heel-toe direction in the foot, to encourage the knee to bend in an appropriate manner based on the orientation of the foot.

Solution times for all the editing and synthesis tasks were similar to the examples in Section 4.7.2.

The resulting edited and synthesized trajectories are shown in the accompanying videos.

4.8 Discussion

Clearly, problem scaling has a very marked effect on solver performance, with the scaled problem achieving a solution several times faster. The objective function is also substantially better for the scaled problem, presumably because the solver fails to make progress in poorly-scaled directions.

For the constraint relaxation method, problems with convergence were encountered as the relaxation parameter α was reduced towards zero. For the walking motion with $\alpha = 10^{-7}$, the solver entered its restoration mode, aimed at restoring feasibility, but was not able to return to a feasible point. The solver was able to recover on the subsequent relaxation step, but required many iterations. For this value of α, violations of the complementarity constraints would be small but not negligible.

For all three test problems, the penalty method performed better in terms of solution time, solver robustness and driving the complementarity error to zero (ζ for the penalty method and α for the relaxation method are directly comparable, both

corresponding to an upper limit on complementarity errors).

The final objective function values for both methods are similar.

Output motions are shown in the accompanying videos. The processed cyclic walk motion satisfies the constraints enforcing physical validity, and retains the desired similarity to the reference. There are still some minor undesirable characteristics in the output motion, such as slight sliding of the feet or failing to distribute forces well between individual contact points, but these may be a limitation of modelling the foot as a single rigid body, rather than an issue with the optimization method.

With motion editing, surprisingly large changes to the original motion are possible before the quality starts to degrade significantly. The feet pass unnaturally close to the ground on the steep slopes, and foot sliding is noticeable for extreme slope and heading angles. These issues are more to do with realism than physical correctness (sliding is allowed in the contact model if the friction force is insufficient to prevent it), and it is likely they could be improved by adding objective terms. For example, Wampler et al. (2014) use an objective function term to penalize large foot velocities close to the ground.

The solver was able to synthesize transitions between animation segments which were of reasonable quality but with some undesirable features, for example sliding of the foot at the beginning of the idle-walk transition. In this case, the animation segments were sufficiently dissimilar that the simple blending approach generated a very poor initialization, and a better initialization strategy could be expected to improve quality.

With some clean-up by an animator, the editing and synthesis tools could reduce the amount of work for tasks such as filling in gaps in motion libraries.

4.8.1 Comparison with Other Work

It is difficult to make rigorous comparisons with other work as no standard test problems have been published and existing work has significant differences in terms of character complexity, duration of motion intervals, simplification of dynamics, usage of spline bases, handling of contacts, choice of objective functions, etc. Bearing these limitations in mind, some comments are presented here in terms of capability and performance relative to previous work.

The method presented here has been demonstrated for physically processing motions, for motion editing and for motion synthesis.

At 25 minutes for a 63 DOF character, the computation time of the method presented here shows a significant improvement relative that reported by Posa et al. (2014), the only previous work with a comparable contact formulation, which took up to an hour to optimize trajectories for a planar character with 13 DOF.

4.9 Summary

This chapter showed that direct trajectory optimization formulations incorporating contacts could be successfully extended to 3D problems. Good problem scaling was shown to have a significant effect on solution performance and quality. The results suggested that the exact penalty method offers better robustness and performance for dealing with the complementarity constraints which contacts introduce.

The presented trajectory optimization method was tested with walking, jogging and running motions. Motion editing was demonstrated, including large alterations to heading direction, slope angle and step cadence. Synthesis of short transitions between existing motions was also shown.

It was shown that trajectory optimization problems can be successfully tackled with interior point solvers as well as previously used SQP methods, and an implementation using non-proprietary software was demonstrated.

Chapter 5

Sampling-based Open Loop Control

5.1 Introduction

This chapter describes the implementation of an open loop control approach which is similar to the work of Liu et al. (2010), but with several significant adaptations. Open loop control applies a control sequence without any feedback to determine if the desired behaviour has been obtained, and so on its own it is useless for interactive applications where disturbances must be dealt with. However, open loop control can be useful when combined with feedback, as it can produce output close to the desired behaviour, leaving the feedback controller with smaller deviations from the ideal trajectory to correct.

In the original paper by Liu et al. (2010), a sampling-based method is used to generate open loop control to reproduce in simulation various contact-rich motions such as walking, running and rolling. For an illustration of the approach, see Figure 2.5.5 and the last paragraph of Section 2.5.8. The basic idea is to segment the motion at boundaries where significant kinematic events occur (such as when leg swing reverses direction), then generate a large number of potential PD control policies for each segment. A control policy consists of PD targets for each DOF for the duration of the segment. Each set of control values leads to a possible end state for the segment simulation, which is evaluated using an objective function based on reproducing the desired reference motion. From the large number of samples, a subset are retained as starting points for simulating the next segment. The retained samples are biased towards better values of the objective function, but some poorer samples are also retained to maintain diversity. A complete path through a number of segments can then be reconstructed by backtracking from the final state. Because the method

is based on forward simulation, it deals well with contact-rich motions, which are difficult to handle with derivative-based methods.

The main difference between the implementation in this project and the original work is the simulation. Liu et al. (2010) use ODE with 2000 time steps per second, which would be too slow to use in a real-time application. Here, 60 time steps per second are used, and simulation is performed using PhysX, for consistency with Chapter 3 and other NaturalMotion software. The sampling-based approach relies on being able to reset the physics state in a reproducible fashion, and the conditions for obtaining consistent results between runs in PhysX are not trivial, as discussed in Section 5.3. More importantly, with much larger simulation time steps, the approach is more sensitive to the quality of the reference motion. At 2000 time steps per second, the original authors report that the method is robust to variations in motion capture procedure and postprocessing, discrepancies between the motion capture actor and simulated character, severe ground penetration, floating and balance issues, retargeting between different characters, and changes in environment such as pebbly ground and ice. However, this robustness may be a result of the very small simulation time step rather than an intrinsic feature of the method, and at 60 time steps per second more physically plausible input motions were found to be required, as discussed in Section 5.4.

In order to generate PD targets for each segment, the original paper samples within a hyperrectangular sampling window, with larger dimensions for the DOFs expected to be more active in the motion. The dimensions for each DOF are chosen manually, and the authors report that one set of sampling window dimensions will work across many example motions, although they do report that manual tuning of the window size is sometimes useful to improve the quality of output motions. Section 5.5 describes a method which allows automatic tuning of the sampling window dimensions, and characterizes the sampling windows with a single size parameter, allowing the effects of the sampling parameters to be elucidated.

As the method is stochastic, and the starting conditions for each segment depend on the previous segment, it is difficult to generate cyclic control sequences. Even if the reference motion repeats, the starting conditions for a segment in one cycle will not match those for the previous cycle, and the control values from the previous segment will not be applicable. No prior method has been demonstrated for generating cyclic policies, and a method for creating a bridging segment which links the end state of the penultimate segment of a cyclic sequence to the start state of the first segment is introduced in Section 5.8.

5.2 Implementation

In this section some details of the implementation used for this project are discussed.

As mentioned in Section 5.1, simulation was performed using PhysX, with 60 time steps per second. A standalone PhysX application was used, although it would be equally possible to integrate the implementation into the morpheme animation engine. The same character model from previous chapters was used, and represented in PhysX using an articulation. The articulation was driven using target joint orientations, which behave as a PD controller. Stiffness and damping values (equivalent to PD gains) for each joint were tuned once to give reasonable outputs. As described by Liu et al. (2010), the samples can be evaluated independently and so the method is highly parallelizable. In this implementation, four simulations were run in parallel on a single machine.

Instead of using the kinematic centroid segmentation method (Jenkins & Mataric 2003) as used by Liu et al. (2010), segmentation of reference motions was performed manually, but employing a similar strategy of segmenting at extremes of motion such as leg swing reversal. The same 64 frame walking motion from Chapter 4 was used as a reference, and divided into 8 segments.

The objective function included the same terms as in the original paper, but with different weights. The main terms in the objective function include a pose term, which measures the discrepancy in joint angles and angular velocities between the simulated character and the target motion, a root term measuring the deviation in root orientation, an end-effector term based on the vertical position of end-effectors, and a balance term, based on the position of the end-effectors relative to the COM, projected onto the ground plane. For full details of the objective function formulation, see the original paper by Liu et al. (2010). The weights used here are shown in Table 5.2.1, along with the weights used for the walking motion in the original paper. The weights used here are mostly proportional to those from the original paper, but a proportionally larger balance weight was found to be useful. Because a different weighting scheme was used, the objective function values are not directly comparable.

Objective Term	Weight (this implementation)	Weight (Liu et al. 2010)
Pose	0.1	5
Root	0.06	3
End-effector	0.6	30
Balance	0.1	10

Table 5.2.1: Objective Function Term Weights

In order to obtain a point in the sampling space to act as the centre of the sampling

window, a preliminary step was run for each motion segment, using CMA (Hansen 2006) to find a point which would lead to a good objective function value. With 1500 CMA iterations and four parallel simulations, this preliminary step required about one minute per motion segment.

5.3 Reproducibility in PhysX

As the sampling method involves constructing the open loop control sequence from a collection of segments, it relies on the simulation behaviour in the concatenated segments being consistent with the behaviour in the individual segments. When the segments are being replayed in sequence, the starting conditions for each segment should be as close as possible to the starting conditions when the segment was originally generated, and the simulation should evolve as closely as possible, i.e. the physics engine should be highly deterministic. Otherwise, the starting conditions for subsequent segments will likely diverge further and further from those expected by the individual segments, and the control values will no longer be appropriate, leading to rapid failure of the open loop control.

The issue of simulation determinism is not discussed in Liu's paper (Liu et al. 2010), and there are several reasons why it may not have been problematic in their implementation. They use ODE for simulation, which is open source and the factors affecting determinism are well known. ODE offers two options for time steps, either an expensive but accurate direct solver or a faster iterative method. In the iterative method, pseudorandom constraint reordering may be used for stability reasons. Determinism can be controlled by using the accurate solver, using a constant seed for the pseudorandom number generation, or disabling constraint reordering. Furthermore, as the control is expressed in terms of PD targets, it does include an element of feedback. In the original paper, the simulation time step is 0.5ms, and the duration of a typical segment is 0.1s, or 200 simulation steps, which may allow some tolerance of small discrepancies in the segment starting conditions.

On the other hand, the implementation here uses PhysX, which is closed source, preventing inspection of the code to determine the factors affecting determinism, and no existing discussion of determinism could be found. Also, a much larger simulation time step is used here, so that each control segment spans only a few simulation time steps, and the feedback component arising from the PD control would be expected to make the open loop control much less robust to departures from determinism.

Therefore, some experiments were performed to investigate what factors influence

determinism in PhysX, and how reproducible the simulation could be made. There are a number of settings in PhysX which might be expected to have an impact on determinism, including:

Restoration Method There are various methods which can be used to restore the PhysX simulation to a previously recorded state. One approach is to use methods exposed in the API to set the transform, velocity and angular velocity of each physics body to their previously recorded values. The problem with this approach is that some internal state not exposed in the API may not be captured and restored. An alternative method is to use the binary serialization tools in PhysX, which allow the binary state to be exported to a chunk of memory and later restored, presumably allowing more internal details of the simulation to be captured, although the documentation does not detail exactly what is stored.

Articulation Iterations In this implementation, the character is represented using a PhysX articulation object. Articulations use an iterative solver to update their state each time step, with separate user adjustable iteration counts for internal driving such as joint forces, and external forces such as gravity and contact forces. If there are any non-deterministic aspects to the iterative solver, such as the order in which constraints are solved, then higher iteration counts would be expected to produce more consistent results between runs.

Resetting Collision State PhysX maintains a cache of collision and contact information for each body in the simulation, to speed up calculations on subsequent frames. The cache can be cleared for a body in the simulation by calling the resetFiltering function, forcing the collision detection for that body to be performed from scratch on the next frame. Clearing the cache may be necessary to avoid invalid information being retained when the state is altered during the process of restoring a previous simulation state. Resetting the collision information causes the next frame to be more expensive, but this is not really a concern since the sampling based method is already an offline approach.

Contact Offset Each body in PhysX has a contact offset, a small extension of its collision volume designed to avoid jittery contacts when objects are at rest. The default value is 0.02 times the typical length scale, which is defined when creating the PhysX scene to reflect the typical size of objects in the simulation (so 1m would be a standard choice for a scene containing a human character, giving a 2cm contact offset). Smaller values can cause jittering problems for situations which are well known to cause simulation difficulties, such as stacks of many objects which are close to stationary, but would not be expected to be problematic for a single character.

In order to test the effect these factors have on the simulation determinism, the

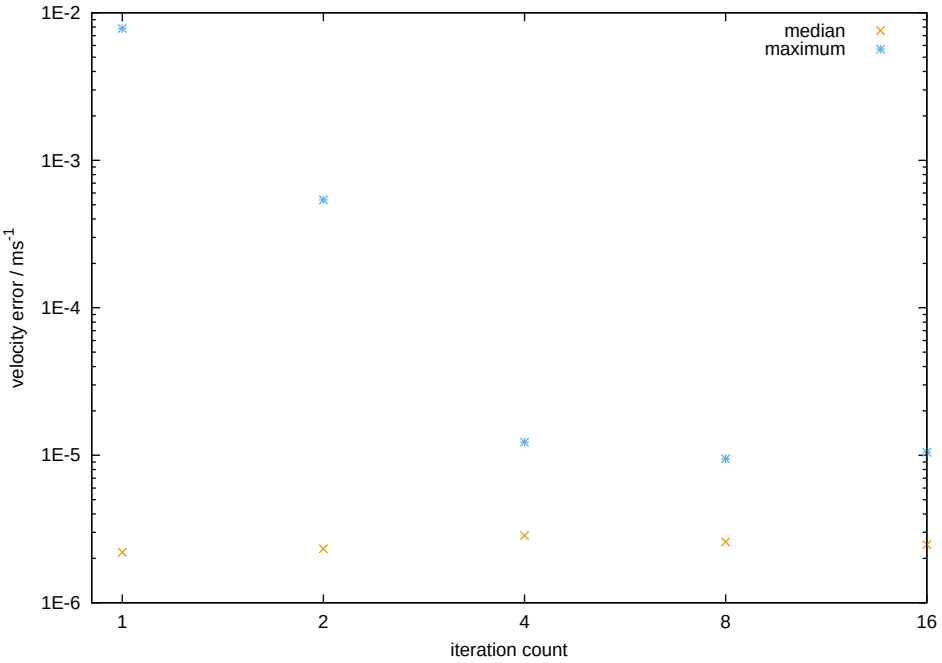

Figure 5.3.1: Effect of Articulation Iteration Count on Run Consistency

simulation was set to a particular state, run for one time step, restored and re-run with identical joint control. The state of the hips body in the character was then compared between the original and repeat runs. This procedure was repeated for each frame in a 64 frame reference animation, and the median and maximum deviations between runs were recorded. The results below are for the discrepancy in the linear velocity of the hips body. The transforms and angular velocities were also compared but showed very similar behaviour to the linear velocity.

The effect of articulation iterations was tested for iteration counts between 1 and 16, increasing in powers of 2, with the internal and external iteration counts both set to the same value. The results for one particular configuration of restoration settings are shown in Figure 5.3.1, but all the other configurations showed similar behaviour. The maximum error can be much larger for very small iteration counts, but no improvement is observed above 4 iterations. Therefore, for simplicity the remainder of this section only discusses results for 4 iterations.

The discrepancy between runs is shown for various configurations of restoration settings in Table 5.3.1. Note that the velocities are reported by PhysX as single precision floating point numbers, with around 7 decimal digits of precision, so an observed difference of 0 only means that the relative error was less than approximately 10^{-7}, not necessarily altogether absent. Evidently using binary serialization and resetting the collision state between runs achieved the best typical deviation between runs, but some frames still showed inconsistent behaviour. However, it was observed that almost all of the frames showed no detectable difference, with the exception of a small number of frames around the time of each new foot contact.

Reducing the value of the contact offset for the character's feet from the default value of 2cm to 0.5mm reduced the discrepancy to unobservable for all the tested frames. It is not obvious why this effect occurs, although one possible explanation is that the contact offset can affect the order in which contacts are resolved.

Restoration Method	Reset Collision State	Median Velocity Error / ms^{-1}	Maximum Velocity Error / ms^{-1}
API	y	2.9E-6	1.2E-5
API	n	7.6E-6	6.4E-4
Serialization	y	0	6.4E-4
Serialization	n	4.6E-6	8.1E-5

Table 5.3.1: Effect of Restoration Configuration on Run Consistency

Based on these results, it appears that it is possible to achieve highly reproducible behaviour in PhysX by using the binary serialization method to restore simulation states, clearing contact data between runs, and using a small value for the contact offset of the feet.

5.4 Input Motion

The previous work on sampling-based open loop control (Liu et al. 2010) reports that the technique deals robustly with reference motion capture data from multiple sources, with differences in capture processes and data postprocessing. They also found that it could handle retargeting artefacts in the input motion, including severe ground penetration and floating, and balance issues resulting from differences in mass distribution between the motion capture actor and simulated character, and was able to cope with retargeting to a significantly different character model, albeit with some reduction in quality.

However, the previous work uses a much higher simulation rate (2000 time steps per second) than is feasible for use in real-time applications, and the robustness may originate to a great extent from the large number of time steps the controller has to compensate for defects in the reference motion or discrepancies between targeted characters. At 60 time steps per second, a typical motion segment only spans a few simulation steps, instead of hundreds, and it is expected that much stricter stability conditions would apply.

The expectation of reduced stability is consistent with observations in other controllers. For example, the generalized biped walking control (Coros et al. 2010) also operates at 2000Hz and is considered robust. Giovanni & Yin (2011) implement that controller in several physics engines, including ODE and PhysX, and find that

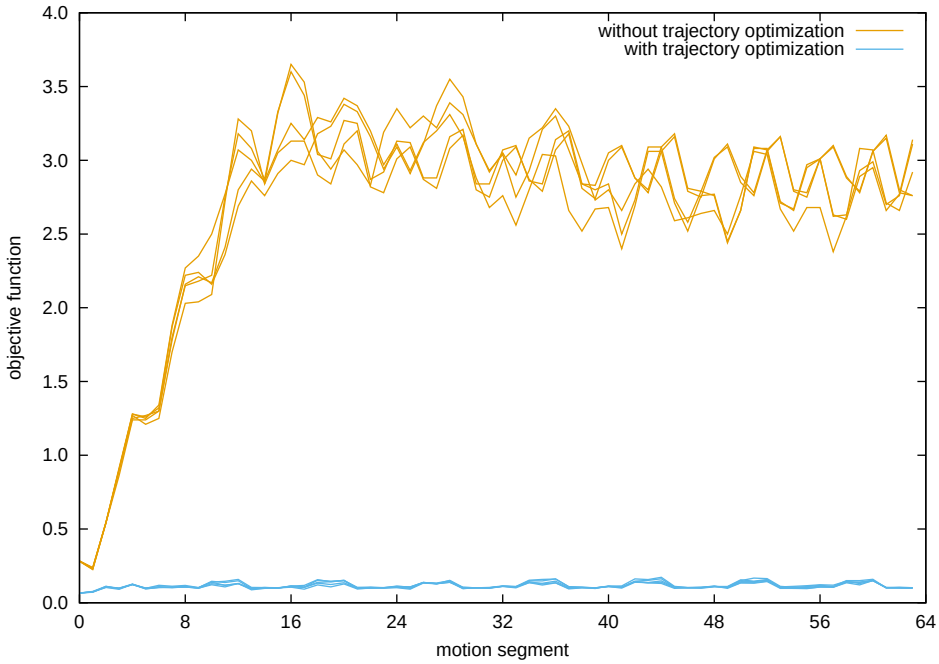

Figure 5.4.1: Sampling Results with Unprocessed Reference Motion and Optimized Trajectory

the controller becomes unstable at time steps above 1-2ms. This result is consistent between simulation software, suggesting that the effect is related to the controller and time step, rather than the choice of physics engine. Similarly, modifying the time step in the SIMBICON source code causes the controller to become unstable below around 750 time steps per second. Chapter 3 describes the difficulties of trying to get a SIMBICON style controller to work with large time steps. The work of Muico et al. (2009), one of the few previous examples using relatively large time steps (120 per second), finds that raw motion capture sequences are poor references and preprocesses target motions using trajectory optimization to improve physical plausibility.

In order to test the sensitivity of the sampling method to the quality of the input motion when using large time steps, sampling was performed for 64 motion segments (8 repeats of the 8 segment walking cycle) using an unprocessed reference motion capture sequence, and also with the output from the same motion processed using the trajectory optimization approach described in Chapter 4. The defects from physical correctness in the raw motion are not particular severe, with typical penetration and floating errors of the order of a few millimetres, and reasonably close attention must be paid for them to be visually noticeable, as can be seen in the videos accompanying Chapter 4. Similarly, visual inspection suggests very little difference between the original and processed motion. Despite the apparently minor difference, when five sampling runs were performed with each input motion, as shown in Figure 5.4.1, the unprocessed motion consistently failed, while the optimized trajectory consistently

produced stable explorations of the control space (the objective function remains relatively small and does not deteriorate from cycle to cycle). Note that the results shown in Figure 5.4.1 relate to a segmented run which has not yet been assembled into a single sequence (see Section 5.6). In order to check that this was not a fluke effect for this particular motion, testing was also performed with several other unprocessed motions, all of which failed to produce valid sequences, and several other outputs from the trajectory optimization, which succeeded except for some motion transformations which were very distant from their original references. This suggests that at 60 time steps per second the sampling technique is very sensitive to the physical plausibility of the reference motion, and that the trajectory optimization approach described in Chapter 4 is successful in generating suitably highly plausible reference motions.

5.5 Sampling Window Tuning and Influence of Sampling Parameters

Liu et al. (2010) draw samples from an n-dimensional hyperrectangular sampling window, where n is the number of DOF, with the size of the window (i.e. the range of target joint angles for each DOF) manually chosen for each dimension based on the expected activity of that DOF. For example, for a walking motion, the sampling window is larger in the directions of the control space corresponding to the hip DOFs than those for the wrist DOFs. Their window dimensions are tuned once and work across various motions, although sometimes retuning can improve output motion quality. The manual window size selection has the disadvantages that there are as many parameters to tune as DOFs, which may not be easy to tune, may vary between motions or even between segments within a motion, and may significantly influence the output quality. Too much sampling freedom might lead to many poor samples, while too little sampling freedom could damage the sampling diversity necessary to achieve far-sighted tracking.

In this section, a method is presented for automatically tuning the relative dimensions of the sampling window for each motion segment as the sampling progresses, based on the distribution of the highest quality samples. The method uses a modification of the CMA algorithm, generating new samples using a covariance matrix calculated from a subset of previous samples with the best objective values, and so automatically extends the sampling window in the directions which are most important for generating good samples. However, instead of shrinking the sample space at each update as is usual for CMA to converge to a solution, the volume of the sampling window is kept constant, to maintain the same level of sampling diversity

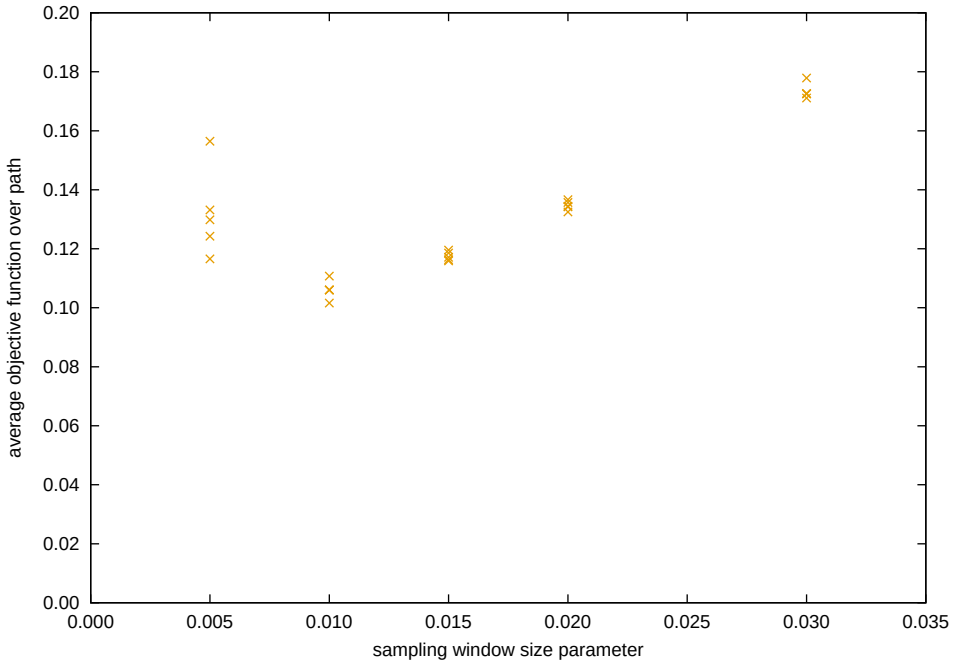

Figure 5.5.1: Effect of Sampling Window Size on Objective Function Averaged over Run

throughout.

As well as avoiding the difficulties with manual sampling windows described above, this allows the size of the sampling window to be characterized by a single parameter, making it easier to examine the effect of sampling parameters on the quality of output motions. The effect of the size on the objective function is shown in Figure 5.5.1. Five sampling runs with 2000 samples per segment were performed at each value of the size parameter, and the average objective function value over each run is shown. The size parameter here is defined so that the sampling window has the same volume as if it were completely symmetrical, with half-width equal to the size parameter in every direction. For a constant value of the size parameter, 0.01 radians, the effect of the number of samples per segment was also investigated as shown in Figure 5.5.2.

The results suggest that the tracking quality of output motions does not depend significantly on the number of samples once a threshold of 1000-2000 samples per segment is reached, but is quite sensitive to the sampling window size with an optimum value around 0.01 for the walking motion considered here. The results are consistent with the expectation that windows which are too large will generate too few high quality samples, while windows which are too small lack the necessary diversity. This convex behaviour and the characterization of the window size using a single parameter make tuning straightforward.

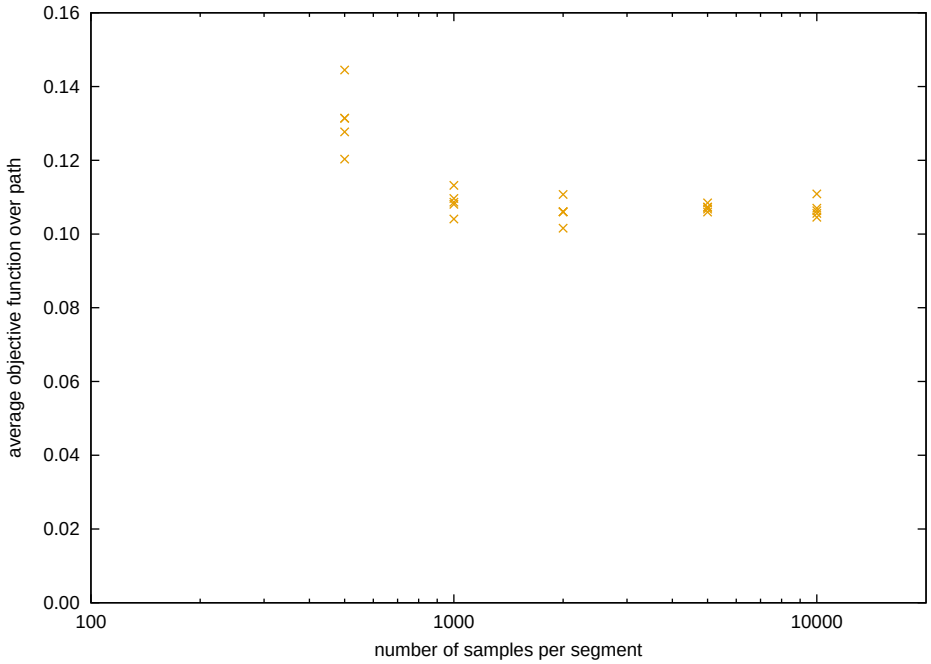

Figure 5.5.2: Effect of Number of Samples on Objective Function Averaged over Run

5.6 Reconstructed Motion

The results up to this point are for individual motion segments computed separately. This section describes the results when the segments are concatenated into a single simulated sequence.

As a measure of stability, the objective function is plotted for a sequence of motion segments in Figure 5.6.1. The output motion is shown in the video accompanying this section. The objective function remains flat for about 50 motion segments, or around 400 frames, before the control begins to become unstable. This suggests that there are still some issues with determinism, as a perfectly deterministic simulation would be expected to exactly reproduce the results from the individual segments and remain stable, as in Figure 5.4.1. To illustrate the effect of one of the settings chosen to improve determinism described in Section 5.3, Figure 5.6.2 shows the difference in objective function values between the segmented and reconstructed path, with and without the contact data being cleared between segment simulations. Without clearing the contact data, differences in the reconstructed path are evident much earlier, and the reconstructed motion starts to fail after around 15 motion segments, rather than 50. Although the optimum conditions from Section 5.3 do not result in perfect determinism, the errors accumulate slowly enough to give a feedback controller a good chance to stabilize the motion.

It was observed that the lateral position of the character's COM could drift quite far from the reference motion, as shown in Figure 5.6.3. This likely occurs because

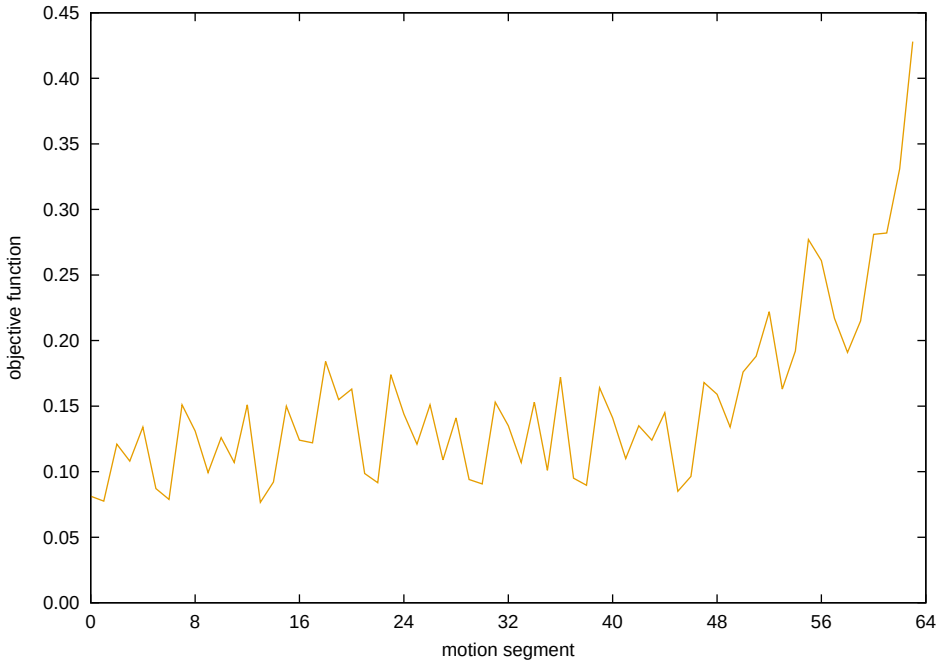

Figure 5.6.1: Objective Function Values for Reconstructed Path

the objective function which was chosen focuses on the orientation of the character's links, and positions of end effectors relative to the COM, and does not contain any terms based on absolute position. This suggests that it might be worthwhile to include some position-based terms in the objective function in future.

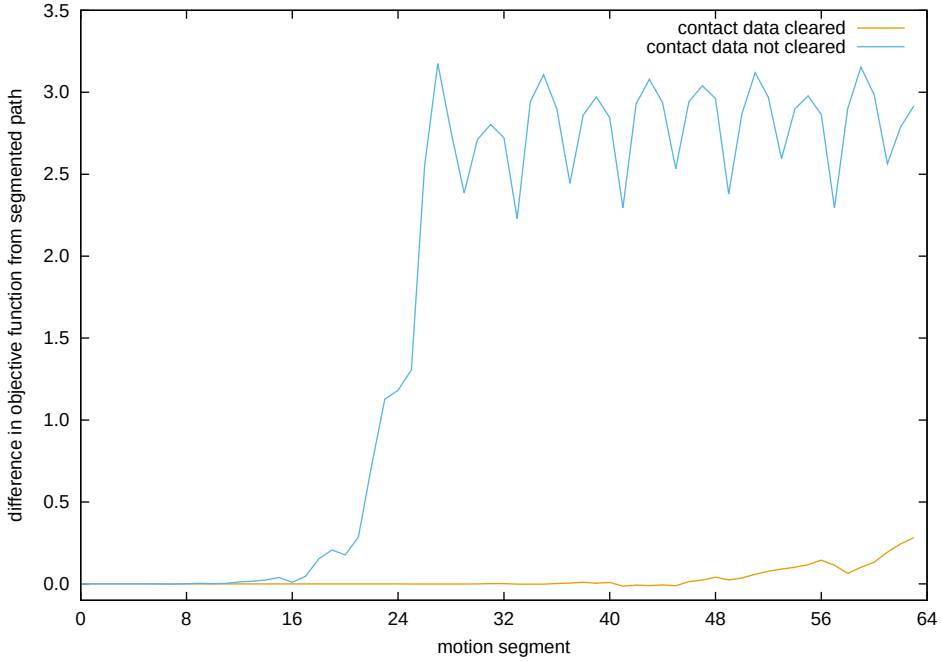

Figure 5.6.2: Effect of Simulation Reproducibility on Reconstructed Motion

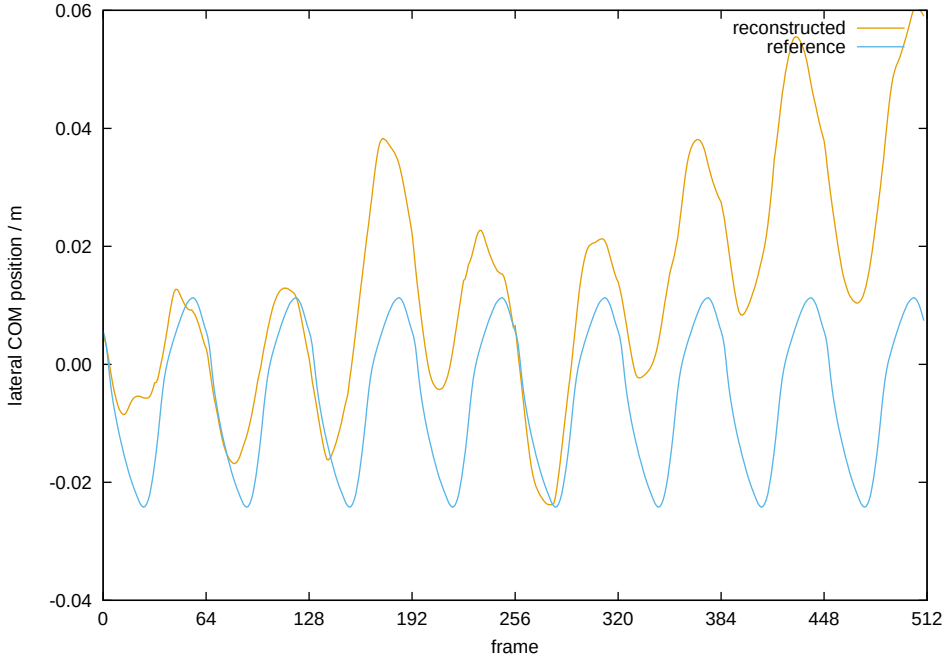

Figure 5.6.3: Lateral Drift in Reconstructed Path

5.7 Performance

For 8 cycles of the walking sequence, or about 8.5 seconds of motion, and using 2000 samples per motion segment, sampling required about 2 minutes on a single 4-core machine. An advantage of using a larger simulation time step is that fewer time steps need to be simulated, so the performance compared well with the original work by Liu et al. (2010), which required slightly more than 2 minutes to reconstruct 5.2 seconds of walking motion using an 80-core cluster.

5.8 Cyclic Open Loop Control

The method described so far is not suitable for creating cyclic control policies, even for cyclic reference motions, due to its stochastic nature. For example, consider the illustration in Figure 5.8.1. The vertical bars indicate the segments on which the reference motion starts repeating. Because the states, represented by circles, are generated using randomly sampled control values starting from the previous segment's state, the state at the beginning of one cycle will not match the state at the beginning of the subsequent cycle, as indicated by the different positions in sampling space of the two red circles. The control values, represented by arrows, are specific to the state from which they were generated, so the control values for one cycle will not be appropriate for the next cycle.

No method has been presented in existing work on sampling-based control to deal with this issue of cycling control policies. Liu et al. (2012), who use sampling-based open loop control together with a feedback controller, do include a method for generating transitions between different motions. However, the transition is achieved using feedback, and the transitions between the open loop controllers themselves are discontinuous, leading to jerky or sometimes failed transitions, even with a small simulation time step.

This section presents a method for generating a bridging segment between two stretches of open loop control segments. It is used here to create cyclic open loop control policies which are stable over several cycles, but could also potentially be used to create transitions between different motions.

As a first step, it is helpful to find a control sequence, of duration equal to the reference cycle, which is already relatively close to looping. As noted in Section 5.6, the lateral deviation from the reference motion could be quite high, which could be a major source of discontinuity between consecutive cycles. For example, the two highlighted points in Figure 5.8.2 occur at the same instant in the reference

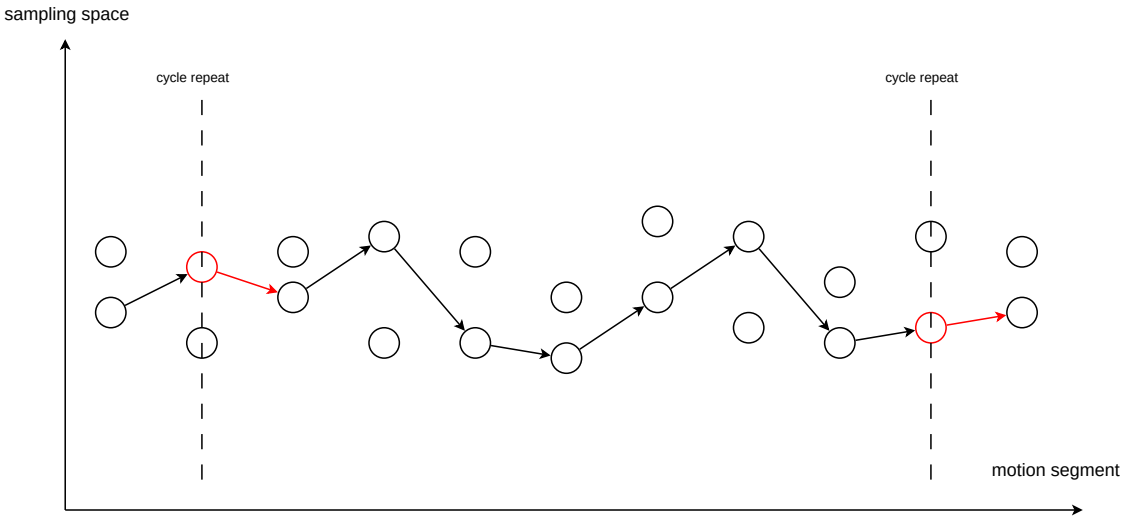

Figure 5.8.1: Differences Between Consecutive Motion Cycles due to Stochastic Approach

cycle, but the large discontinuity between them would make it difficult to construct a cyclic motion based on that stretch of reconstructed motion.

For an example reconstructed motion, Figure 5.8.3 shows the magnitude of the discontinuity in lateral position between each segment and the corresponding segment in the previous cycle. It also shows, for each segment, the root mean square deviation from the reference motion, taken over the duration of the cycle ending on that segment. The second measure is useful for ensuring that the amplitude of the lateral motion closely matches the reference motion. Segments for which both values are small, such as indicated by the vertical line in Figure 5.8.3, indicate the end points of stretches of motion which are more favourable starting points for generating cycling behaviour.

Once a stretch of reconstructed motion that is not too far from cyclic is identified, the following approach can be used to construct a cyclic control sequence. The basic idea, as illustrated in Figure 5.8.4, is to optimize control values for a bridging segment, such that repeating the control values for all subsequent cycles leads to stable control. The selected initial motion sequence is already reasonably close to being cyclic, as indicated by the control arrow almost joining up with the state from the previous cycle. The hope is that it will be possible to find control values for the bridging segment which bring the subsequent state close enough to the cycle's beginning state to be within the region of stability for the repeated control sequence.

In the implementation here, CMA was used as follows to determine suitable bridging control values, with evaluation based on the average objective function value over a series of cycles. During each cycle, the control values generated by the current CMA iteration were applied during the bridging segment, with all the other segments repeating the control values from the original, almost cyclic, control sequence. The

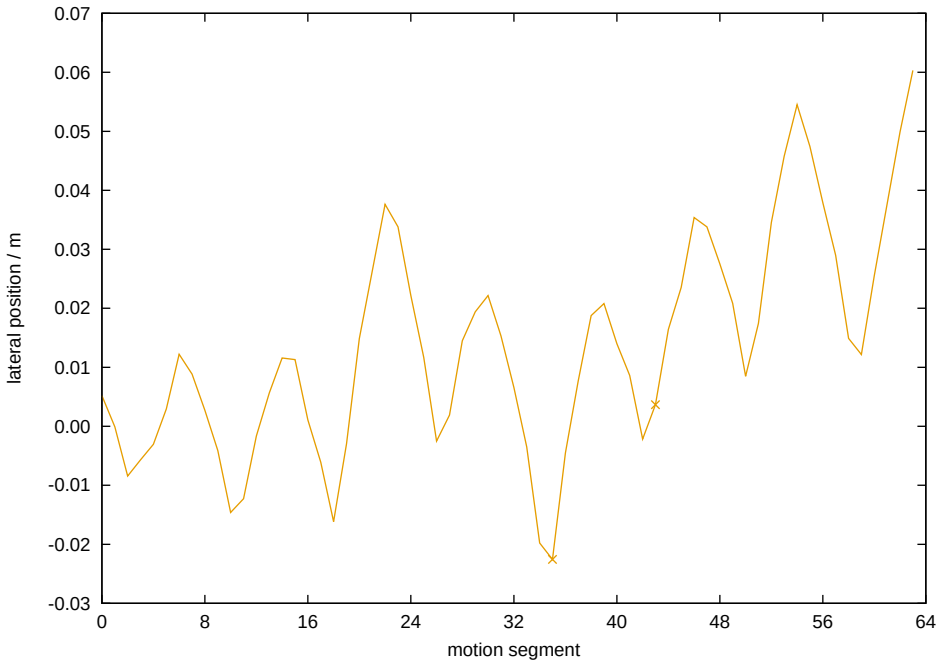

Figure 5.8.2: Illustration of Poor Cyclic Behaviour

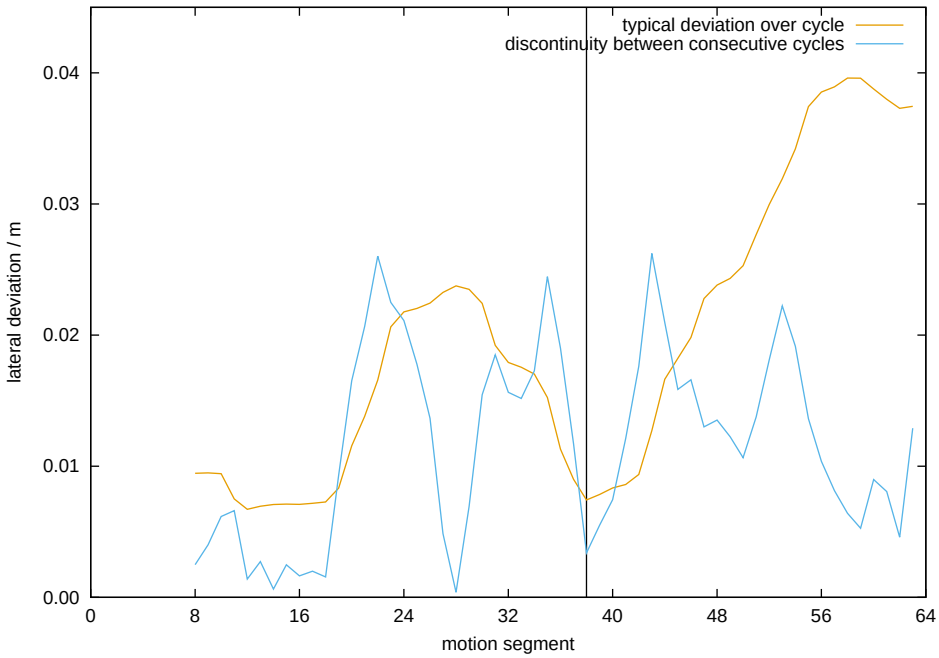

Figure 5.8.3: Selection of Motion Segment with Small Discontinuity Between Cycles

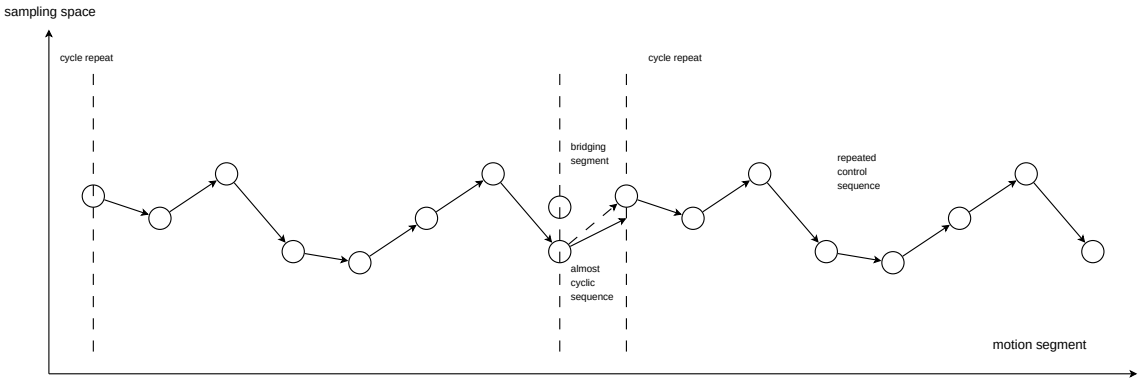

Figure 5.8.4: Bridging Strategy

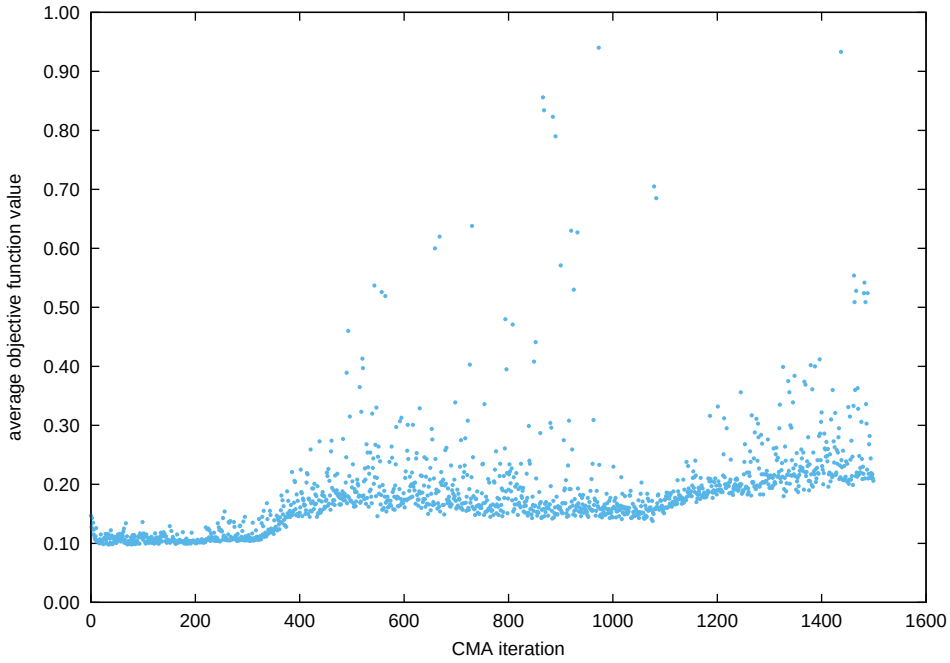

Figure 5.8.5: Objective Function Values During CMA Optimization of Cyclic Policy

duration of the evaluation was extended as the CMA run progressed, from 1.5 to 3.5 motion cycles, in order to start with an easier problem and then gradually place more emphasis on longer term stability. 1500 CMA iterations were used, requiring a total time of about 18 minutes, although the implementation only used a single simulation and performance could easily be improved using parallel simulations.

The results of the CMA run are shown in Figure 5.8.5. The objective function deteriorates somewhat as the evaluation duration increases through the run, indicating that the method does not produce completely stable cyclic control. However, it does get worse relatively slowly, suggesting that it may still stay stable on a long enough time scale to be useful.

The objective function for a path reconstructed using cyclic control values is shown in Figure 5.8.6, and the output motion is shown in the video accompanying this section. The reconstructed path does not show the motion prior to the first bridging segment,

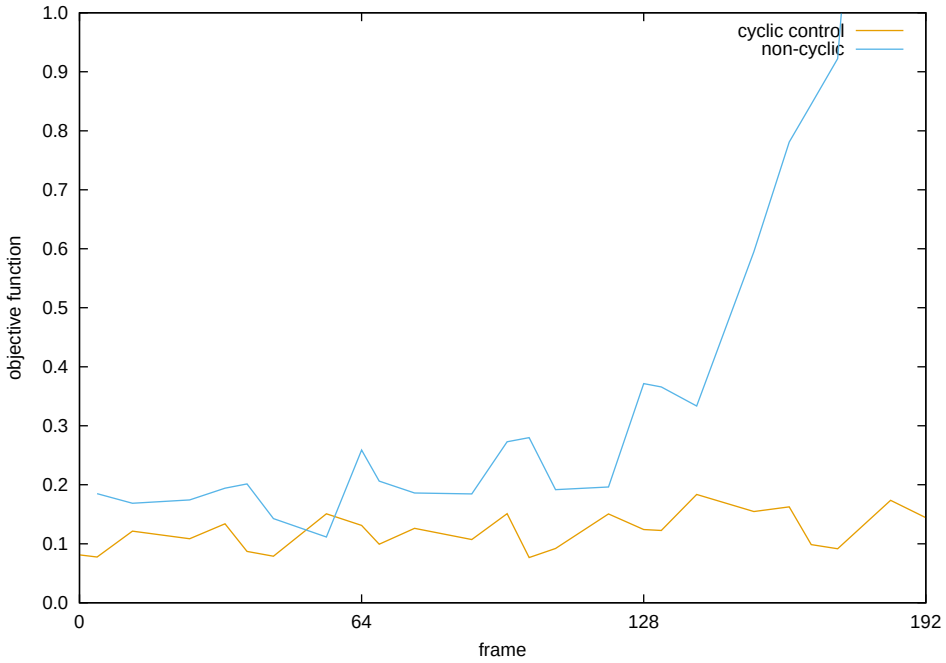

Figure 5.8.6: Objective Function for Reconstructed Path with Cyclic Control

so the motion is actually stable for approximately one cycle longer than Figure 5.8.6 suggests, or about 200 frames altogether. Thus, the cyclic control policy is less stable than the non-cyclic policy (which started to fail after around 400 frames) but may be more favourable as the basis of feedback control as it eliminates the problem of dealing with one large discontinuity per cycle, leaving an error which accumulates over multiple cycles.

5.9 Summary

This chapter described the implementation of a sampling-based approach for generating open loop control policies which reproduce reference motions. The approach was based on work by Liu et al. (2010) but required adaptations to reduce the simulation rate from 2000 time steps per second to 60, and to use PhysX, a common game simulation engine.

The sampling-based method relies on the determinism of the physics simulation, and some care was required to obtain deterministic behaviour using PhysX. Various factors were found to affect reproducibility in PhysX, including the use of serialization or other methods to restore the character's state, the number of solver iterations, internally cached information about contacts, and the size of the region surrounding objects in which contacts could be detected. With suitable choices for these settings, it was possible to obtain highly deterministic behaviour, but simulations with nominally identical starting conditions would still diverge after a few hundred time

steps.

Unlike the original work by Liu et al. (2010), which was robust to imperfections in the input motion, the large time step implementation was found to be susceptible to failure using reference motions with even quite subtle departures from physical plausibility. However, using the trajectory optimization approach from Chapter 4, it was possible to generate visually similar reference motions which allowed the sampling method to succeed.

A technique was introduced to automatically tune the dimensions of the sampling windows used for each motion segment. Experimentation suggested that the tracking quality of the motion output was more sensitive to the sampling window dimensions than the number of samples per motion segment, with no observable benefit above 2000 samples per segment.

Relatively poor tracking of the lateral COM position was noted, likely due to a lack of bulk position criteria in the objective function used.

The sampling-based method inherently produces non-cyclic control policies. A method was presented here to attempt to create cyclic policies, based on searching an interval of many cycles for a stretch of control values which minimized the cyclic discontinuity, then creating a bridging segment using an evolutionary method. With this approach, it was possible to generate cyclic control policies which avoided large discontinuities between cycles, but with stability reduced to only around 3 walking cycles. Despite the limited stability, it is expected to be easier for a feedback controller to compensate for the resulting slowly accumulating errors than to deal with a large discontinuity between cycles.

Chapter 6

Feedback Control

6.1 Introduction

Chapter 5 described the development of an open loop controller using 60 simulation time steps per second. Open loop control is very little use on its own however, as controllers for interactive applications must be able to deal with unexpected disturbances. Indeed the controllers developed in Chapter 5 fail after a few cycles even without disturbances, due to accumulation of errors and the difficulties of creating cyclic control policies using sampling-based methods. Therefore, this chapter describes the development of feedback policies with the aim of stabilizing the open loop controller. The approach used here is similar to previous work by Ding et al. (2012), as described in Section 6.2. However, as with Chapter 5, the large increase in time step results in significant differences in the conditions required for stable control, as described in Section 6.3. Section 6.4 examines how robust the resulting feedback controllers are to external forces. Computational performance relative to the requirements of real-time applications is discussed in Section 6.5. Section 6.6 describes the use of machine learning techniques to predict whether the controller will remain stable given the character's current state, which has many potential applications in high level control.

6.2 Related Work

The feedback approach used here is similar to the method described by Ding et al. (2012), which uses CMA evolutionary optimization (Hansen 2006) to learn linear feedback policies which map a deviation from the desired state, ds, to an adjustment

to the control action, da, using a feedback matrix M

$$da = Mds \tag{6.1}$$

where s is an n-dimensional vector of state and sensory information expected to be useful in deriving feedback laws, and a is an m-dimensional vector spanning the space of actions expected to be most relevant to the feedback control. The feedback matrix is therefore $m \times n$. The method assumes an open loop control policy is already available, and Ding et al. (2012) use the sampling-based open loop control approach introduced by Liu et al. (2010) for several example motions. The elements of the feedback matrix are then used as the parameters in the CMA optimization. Each member of the CMA population is evaluated by a policy rollout, that is by applying its parameter values to the feedback policy and performing simulation for some duration, and measuring tracking quality or other desirable features using an objective function. Due to the dimensionality and irregularity of the search space, some problems are difficult to optimize directly, but may be addressed by starting with simpler problems and progressively adding difficulty.

To encourage sparsity in the control formulation, the feedback matrix may be decomposed into the product of $m \times r$ and $r \times n$ components, with r small compared to m and n so that the total number of parameters, $r \times (m + n)$, is reduced. Values of r up to 3 are used for the example motions. Sparsity may be further encouraged by an objective function term which rewards zero-valued elements in the feedback matrices.

The feedback decomposition approach also has the advantage that the component matrices have the effect of automatically identifying low-dimensional state abstractions and low-dimensional co-ordinated actions, and therefore the method does not rely on the motion-specific expertise normally needed to manually select such low-dimensional spaces. It is also shown that information such as ground reaction forces, which is not commonly included in state descriptions, can contribute to effective feedback policies.

As well as avoiding dense feedback structures, Ding et al. (2012) do not consider feedback policies which depend on motion phases as they would result in large parameter counts.

The simulation time step is not stated, although the computation times reported suggest that it is probably close to 0.5ms, which is very commonly used in other work. With long policy rollout durations, up to 50s, and single threaded simulation, some optimizations require hundreds of hours to complete, although it is noted that

the simulation for the individuals in the CMA population is trivially parallelizable.

Liu et al. (2012) use a similar approach to learn feedback controllers for running, with transitions in and out of several highly dynamic obstacle clearing motions. An affine structure is used for the feedback, rather than linear, to allow for parameterization of the motions. The running motion is parameterized in terms of speed and direction, and the other motions are parameterized in terms of features such as obstacle height. For the running motion, a 12-dimensional state vector and 9-dimensional action vector are used. The order reduction approach suggested by Ding et al. (2012) is used, with $r = 3$, so the total number of parameters for running is $3 \times (12 + 9) = 63$.

Simulation is performed using ODE, with a time step of 0.5ms. Compared to the work by Ding et al. (2012), shorter policy rollouts are used (up to 20 cycles of running) and early termination of simulations is used to reduce the evaluation cost for policies which fail rapidly. Multiple simulations are also run in parallel on a 24-core machine, with the running motion requiring 11 hours of computation. Note that this includes motion parameterization to different speeds and directions.

Although Liu et al. (2012) split some of the obstacle manoeuvres into separate motion phases for feedback control, the running controller uses static feedback throughout the motion, as with the walking motion generated by Ding et al. (2012). There are however reasons to consider including phase-dependent feedback. In human locomotion, reflexes, which contribute to feedback control, are known to be phase-dependent (Zehr et al. 1998). Several simulated control strategies also employ phase-dependent feedback, for example Luksch et al. (2010), Geyer & Herr (2010), and Wang et al. (2012).

Machine learning techniques are also used here to predict whether a feedback controller can maintain stability given the character's current state. In related work, Faloutsos (2002) uses support vector machines (SVM) in order to predict whether transitions between different controllers will be successful. SVMs are a supervised learning approach, that is they learn based on a set of training data which has already been classified, in this case a set of vectors describing the character state with labels indicating transition success or failure. Training the SVM model may take considerable time, but it only needs to be done once, and can be performed offline. Prediction is performed online, so the prediction time and amount of memory required to store the model are important considerations.

SVMs work by mapping the input space to a higher-dimensional space in which the data are more separable based on their labels. The mapping may use a linear or nonlinear kernel, with common nonlinear kernels including polynomials and radial basis functions (RBF). Within the high-dimensional space, a hyperplane is sought

which balances two desirable properties. The first desired outcome is to create the largest separation between the classification regions, while the second is to minimize incorrectly classified examples. Training points which are closest to the classification boundary are called support vectors. In order to make predictions for new data points, not included in the training set, it is necessary to determine which side of the hyperplane they fall in the high-dimensional space. The cost of this prediction is proportional to the number of support vectors, which can grow linearly with the size of the training set.

For the conditions considered by Faloutsos (2002), polynomial kernels produce better accuracy than RBF, with accuracies ranging from 70-100%, depending on the transitions considered. For typical training set sizes of around 10000 examples, training times range from a few minutes to a few hours, depending on which kernel is used, with prediction times on the order of milliseconds. The memory requirements for the models are not reported.

As noted above, the prediction costs for standard SVM scale with the number of support vectors, which can be problematic for large training sets. Jose et al. (2013) address this problem by introducing local deep kernel learning (LDKL), using local kernels with a tree-based structure. The tree structure means that the prediction cost scales logarithmically, allowing prediction costs 2 or 3 orders of magnitude smaller, although with a modest reduction in accuracy for the examples they consider.

6.3 Large Time Step Feedback

This section describes the feedback implementation used here, using PhysX with 60 simulation time steps per second, as in previous chapters. Again, due to the much larger time steps, the conditions for stability are expected to be stricter.

The cyclic formulation described in Section 5.8 was used as the underlying open loop control, using the same 64 frame walking motion as previous examples.

A state vector was manually selected based on the root orientation and height, the horizontal COM velocity and the positions of the feet, a total of 12 dimensions. Ground reaction forces, which were successfully used as state features by Ding et al. (2012), were considered, but found to be too noisy to be useful. For the action space, the joint target DOFs for the left and right hip, knee and ankle, and the lower joint of the torso were used, a total of 21 dimensions.

The feedback matrix elements were optimized using CMA (Hansen 2006), with a

population size of 24. At each CMA iteration, each member of the population was evaluated by applying its values to the feedback policy, and performing simulation with those values. Each frame of simulation was compared to the corresponding frame of the target motion using the same objective function as in Chapter 5, and the objective function values were averaged over the whole simulation duration to give an overall objective value.

In order to avoid training controllers which were overly specific to particular testing conditions, a number of simulations, with different starting conditions, were run for each trial feedback policy and the objective function was also averaged over the separate runs. For the results described in this secion, the testing conditions differed only in terms of the point in the motion cycle at which simulation was initiated. Repeated testing is more important in Section 6.4, which also involves external disturbing forces, but was also included here as a common implementation was used for testing with and without disturbances.

Based on the results regarding the effect of the number of evaluation conditions on controller robustness reported by Wang et al. (2010), each trial policy was evaluated across 10 runs. As with the method of Wang et al. (2010), the starting conditions were randomly determined, but the same conditions were used for each member of the CMA population, to make the evaluation fair across the population. New starting conditions were randomly determined every 10 CMA iterations.

In keeping with the common strategy of aiding CMA convergence by starting with a simpler problem and incrementally adding complexity, the interval over which the policies were evaluated was increased from 1.5 motion cycles to 4 as the optimization progressed.

A total of 1500 CMA iterations were used for each optimization. With 4 simulations run in parallel on a single machine, optimization required 3-4 hours depending on parameter count (see below).

Controller optimization was tested for various feedback structures, both full and reduced-order, and with and without phase-dependence. Reduced-order controllers used the matrix decomposition approach described by Ding et al. (2012), with $r = 3$. Motion phase was defined based on the contact states of the feet, since feedback strategies involving manipulating GRF would be expected to depend significantly on which contacts are available. For the purpose of determining phase, each foot was considered to be in contact if PhysX reported any contact forces on the previous simulation frame. If neither foot reported contacts, the foot closest to the ground would be considered to be in contact. Two double stance phases were defined, based on which foot was further forward, as well as two single stance phases, for a total of

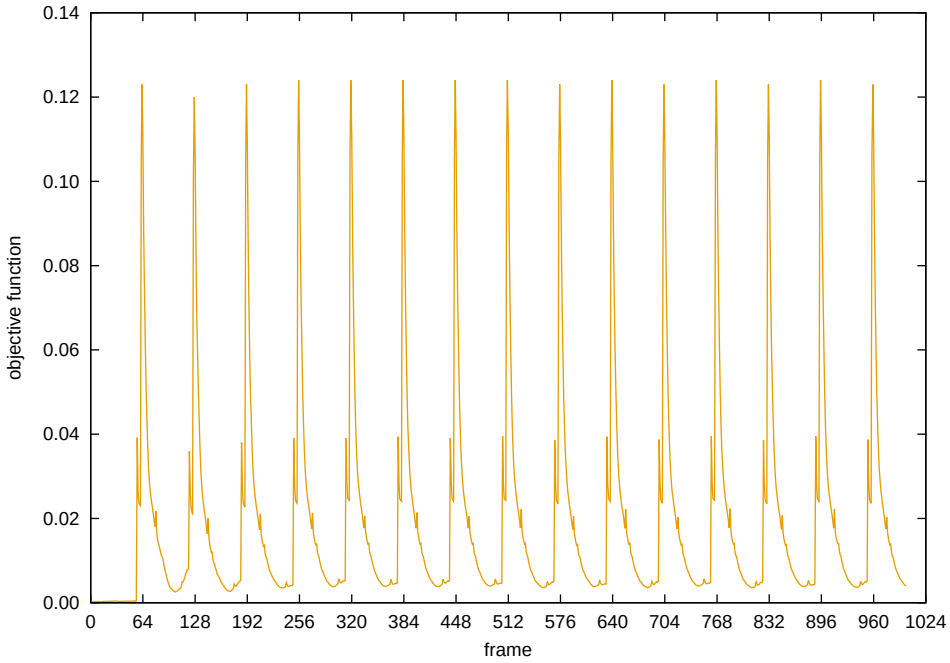

Figure 6.3.1: Progression of Objective Function for Successful Controller

4 phases. It might be possible to exploit symmetry between the left and right stance phases to reduce the number of phases to 2, but this option was not explored.

Results for the different feedback structures are shown in Table 6.3.1. Only using full order, phase-dependent feedback resulted in a successful controller, which remained stable after testing for 1000 simulation time steps, as shown in Figure 6.3.1. The periodic spikes in objective function occur because of the difficulty of bridging the discontinuity between the two ends of the motion sequence to make a cyclic control policy. The output motion is also shown in an accompanying video. All the other tested configurations led to rapid failure, which probably indicates that the CMA optimization had completely failed to converge. This suggests that for large time steps, the low-dimensional abstracts which reduced-order feedback creates are too sparse to describe adequate feedback policies, and that single phase policies cannot capture the adaptations in feedback strategy necessary to deal with different contact states.

Controller Type	Phases	Parameters	Result
reduced order	1	99	failure
reduced order	4	396	failure
full order	1	252	failure
full order	4	1008	success

Table 6.3.1: Effect of Feedback Configuration

Despite the large parameter count for the full order, multiphase controller, no difficulties were observed with convergence or overfitting. However, each iteration of the CMA algorithm does include an eigendecomposition step, with cost cubic in

the number of parameters. For the two cases with the largest parameter counts, average computation times for the CMA update and for the iteration as a whole were measured, as shown in Table 6.3.2. Timings were measured at the start of the optimization, before the simulation duration had been increased, so later iterations would have higher simulation costs. For 396 parameters, the CMA cost is still very small compared to simulation, but it is more significant with 1008 parameters, requiring about one third of the total computation. No special effort was made to make the CMA implementation used here efficient, but it might be necessary to consider performance of the CMA update if a larger parameter count were used, or if simulation were parallelized across more than 4 threads.

Parameters	CMA Update Time / s	Total Time per Iteration / s
396	0.088	2.33
1008	1.132	3.55

Table 6.3.2: Computation Times for CMA Update

6.4 Robustness to Disturbances

In order to evaluate the robustness of the successful feedback controller, the optimization was repeated, this time with random external forces applied during the training process. Apart from the introduction of disturbances, the optimization process was identical to that described in Section 6.3.

During training, forces were applied to the topmost link in the torso, roughly corresponding to the character's chest, for a duration of 0.1 seconds (6 time steps), beginning at a random point in the motion cycle. The forces were parallel to the ground plane, with components in the two horizontal directions drawn from independent normal distributions with standard deviation σ_f. This means that the squared magnitude of the overall force has a χ^2_2 distribution. The force magnitude is distributed as shown in Figure 6.4.1, with median value approximately $1.18\sigma_f$, as illustrated by the vertical line. It also means that the direction of the forces is uniformly distributed. This distribution was chosen to ensure the controller was tested with a wide range of force magnitudes, to avoid training that was overly specific to a particular level of disturbances.

A random horizontal displacement was also applied to the application point for each force, so that disturbances would also include a rotational component. The displacement in each horizontal direction was randomly drawn from a normal distribution with standard deviation of 0.1m.

As with Section 6.3, a total of 10 testing conditions were randomly determined

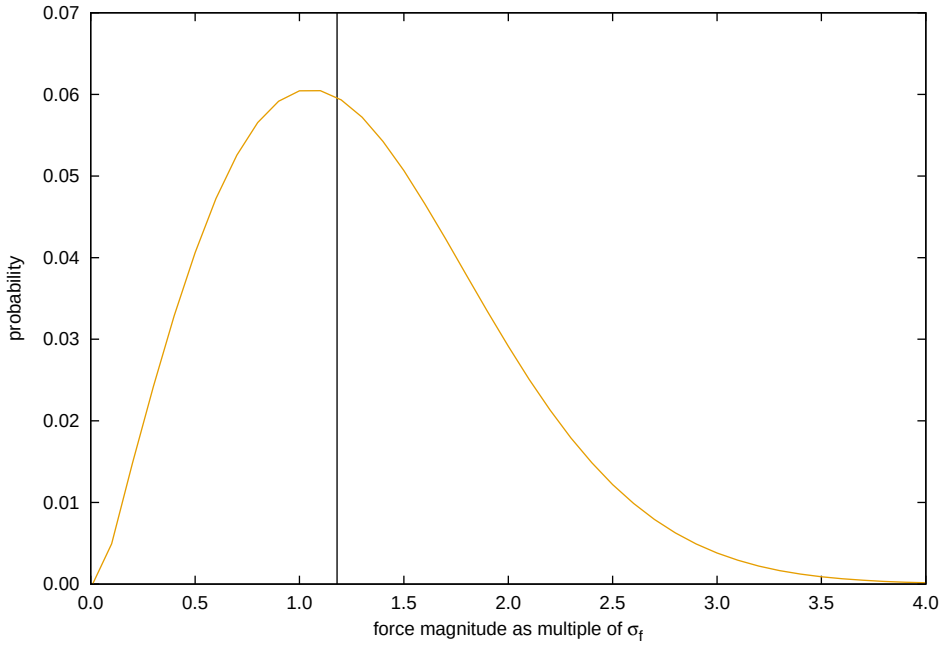

Figure 6.4.1: Distribution of Force Magnitudes, Relative to σ_f

after each 10 CMA iterations, and applied identically to each member of the CMA population.

A series of optimizations was run, starting with a small value for σ_f, and increasing it each time, with each run initialized using the controller from the previous optimization. The robustness of each resulting controller was evaluated by running a series of 10000 trials with random force magnitudes, directions and timing. Each trial was considered successful if the character was upright and close to its target pose 10s after the disturbance was applied. The trial forces were divided into bins of size 50N, and the proportion of successful trials was calculated for each bin. Controller robustness increased up to $\sigma_f = 200N$, after which robustness began to deteriorate, as shown in Figure 6.4.2. This is consistent with expectation, as values of σ_f which are too large will lead to many failures during training, which do not provide much useful information to the optimization.

For the controller trained at $\sigma_f = 200N$, the robustness results were measured separately depending on the direction from which the force acted, split into four quadrants. As shown in Figure 6.4.3, the controller was a good deal more robust to disturbances from behind, with relatively little difference between the other three quadrants.

For comparison, the robustness results for previous work are shown in Table 6.4.1. It is difficult to make a rigorous comparison, as previous results vary in testing conditions. For example, some work applies disturbances only at fixed times in the motion cycle, while other results use arbitrary timing. Differences also exist in the

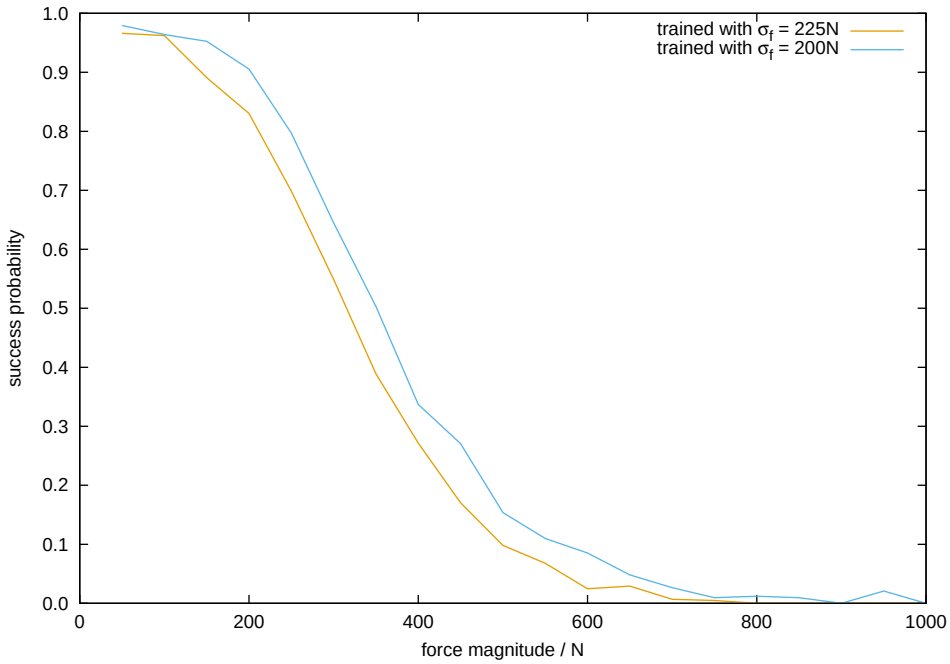

Figure 6.4.2: Robustness Starting to Deteriorate as Training σ_f Becomes Too Large

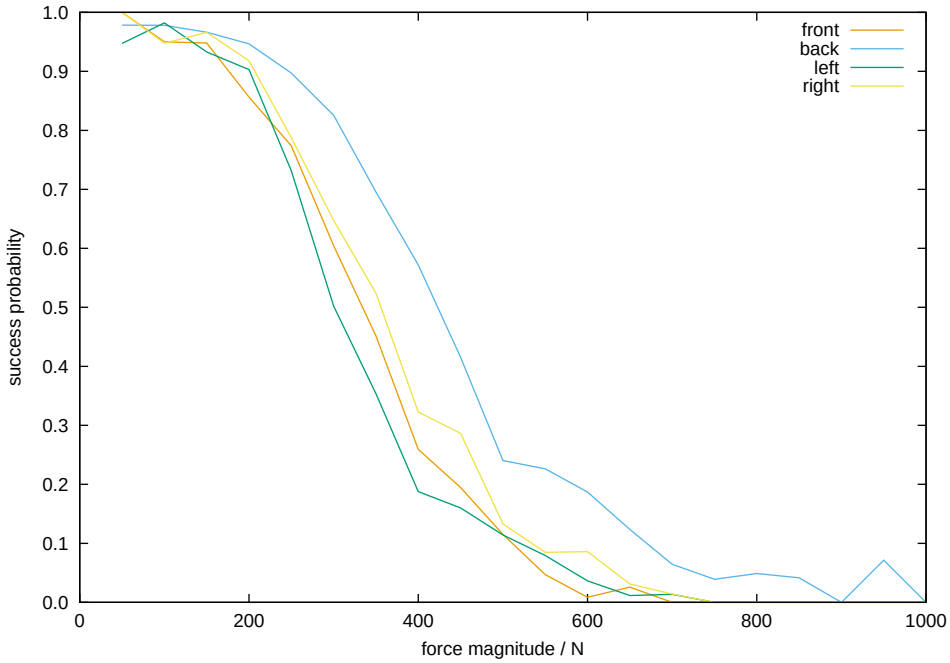

Figure 6.4.3: Robustness by Force Direction for Controller Trained with $\sigma_f = 200N$

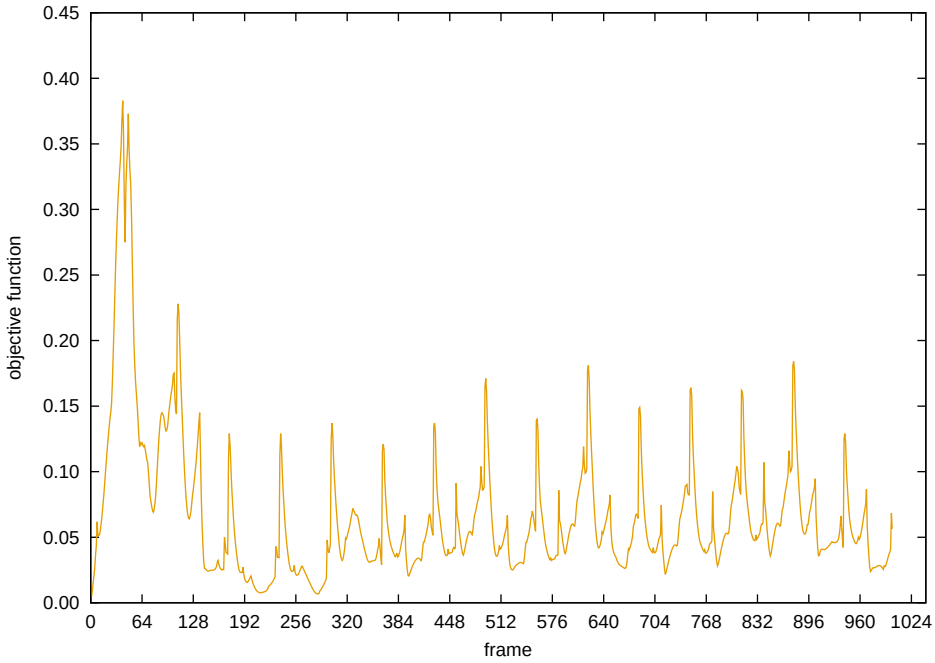

Figure 6.4.4: Progression of Objective Function Following Disturbance of 185N for 0.1s

durations of impulses and distribution of directions. The previous work typically reports the largest forces which the controller was able to recover from, which is not an ideal measure at it could depend on the number of tests performed. For example, the controller described here was able to recover from forces as large as 920N (an impulse of 92Ns), but only with very low probability, so it might not have been observed if a smaller number of trials had been performed. With these limitations, only a qualitative comparison is possible, but the controller developed here appears to be in the same range of robustness as much of the previous work, although the most robust controllers are able to recover from impulses several times larger. However, the robustness is suprisingly good, considering that most previous results are for much smaller simulation time steps. Only Muico et al. (2011) use a similarly large time step, but they use a more expensive control scheme (see Section 6.5, and require a composition of multiple controllers to achieve their highest robustness results.

Figure 6.4.4 shows the behaviour of the objective function following the successful recovery from a force of 185N. The robustness of the feedback controller is also illustrated in an accompanying video, which shows a series of successful recoveries.

Reference	Force and Duration	Maximum Impulse / Ns	Time Steps per Second	Notes
Yin et al. (2007)	340N, 0.4s	136	2000	fixed phase, applied just after foot contact
Wang et al. (2010)	475N, 0.4s	190	2400	trained with arbitrary timing during cycle
Wang et al. (2009)	200N, 0.4s	80	2400	every 4 seconds for 40 seconds, phase not stated
Lee et al. (2010)	160N, 0.4s	64	900	every 4 seconds for 40 seconds, phase not stated
Wang et al. (2012)	100N, 0.4s	40	2400	every 4 seconds for 40 seconds, phase not stated
Ding et al. (2012)	203N, 0.4s	81	not stated	
Ding et al. (2012)	500N, 0.2s	100	not stated	
Muico et al. (2011)	150N, 0.5s (approx)	75	120	applied for about one half-cycle around double-support phase; composite controller
Coros et al. (2010)	600N, 0.1s	60	2000	arbitrary timing during cycle
Wu & Popović (2010)	350N, 0.1s	35	2048	
Wu & Zordan (2010)	170N, 0.1s	17	2000	

Table 6.4.1: Robustness Comparison of Previous Work

6.5 Controller Performance

In previous work, character controllers are typically described as real-time if they require no more than one second of CPU time to produce one second of simulated motion. However, most practical real-time applications will involve many other processes competing for CPU resources, and a budget of more than 5-10% of the CPU time for character simulation would likely be infeasible. To avoid dropping frames, it is also desirable for computing costs to be very consistent between frames.

The controller developed here was timed over a series of several hundred time steps, and found to use an average of 0.62ms per step, with a standard deviation of 0.08ms, for both computing the control policy and performing the simulation. At 60 simulation time steps per second, this amounts to less than 4% of a single CPU core's time. Almost all previous work uses much smaller time steps and requires at least an order of magnitude more computing time. The only work known to be comparable in performance is by Muico and colleagues (Muico et al. 2009, 2011), but for their walking controller they report an average time of 1.6ms per time step, with 120 steps per second, which is several times slower. They also have a much more complex control scheme, and computing costs which are highly variable between frames, varying between 1.0 and 4.1ms. The cost and variability are dominated by the LCP solver used to deal with contacts, which they describe as overly precise. It is possible that faster and more consistent times could be achieved with a more efficient LCP solver, but this has not been demonstrated. Therefore, with the possible exception of Muico's work, the work described here is believed to be the first locomotion controller with sufficiently low computation cost to be practically used in a real-time application.

6.6 Machine Learning for Prediction of Controller Success

This section describes the use of machine learning to predict whether or not a feedback controller will remain stable, based on the current state of the character. There are many reasons why such predictions could be useful. As described by Faloutsos (2002), predictions could be used to mediate transitions from one controller to another as dictated by high level control. Secondly, a higher level of robustness could be achieved if multiple feedback controllers are trained for a particular motion sequence, and prediction is used to select a feedback policy which is likely to succeed. This is similar to the idea of composite controllers employed by Muico et al. (2011). Similarly, it may also improve robustness to reset the timing of the open loop control

policy after certain disturbances, which could be achieved by performing prediction for a range of instants surrounding the current control instant, and searching for more stable timing. Finally, early failure prediction could be useful in real-time applications such as games, where failure of simulated characters can be very damaging to immersion, and would allow the failure to be handled more gracefully, for instance by switching to a recovery controller or falling back to a kinematic method.

Previous work (Faloutsos 2002) uses support vector machines (SVM), with prediction times in the order of milliseconds. Such prediction times are significant compared to the control and simulation cost reported in Section 6.5, which means that strategies relying on multiple predictions, such as the increased robustness methods described above, or transition between a dense space of controllers, would be infeasible under real-time performance constraints. Here, a standard SVM approach similar to that used by Faloutsos (2002) is compared with a more recent variant of SVM (Jose et al. 2013), which uses a tree-based kernel structure to improve prediction times.

In order to generate training data, 1000 trials were performed using the feedback controller, trained with $\sigma_f = 200N$, subjected to a random force for 0.1s and then continuing to simulate for 10 seconds (600 time steps). At the end of each trial, success was defined by remaining upright and close to the target pose. All the preceding time steps in the trial were then marked as leading to success or failure accordingly. A vector representing the state of the character, along with the success label, constituted one example in the training data. To keep the size of the training set manageable, the state at each time step was included in the data set with probability 0.1, leading to a total training set size of approximately 60000. In order to achieve balance between positive and negative examples in the training data, trials were conducted with $\sigma_f = 300N$, which led to a success rate of approximately 50%. The same procedure was used to generate a set of test data, again with around 60000 examples, to use in evaluating the trained models.

Two different representations of the character state were tested. The first used the 12 features included in the feedback policy state vector (the root orientation and height, the horizontal COM velocity and the positions of the feet), along with the current timing in the motion cycle, for a total of 13 features. A higher dimensional state vector was also tested, to examine whether a more detailed state description would lead to more predictive power. The second state vector included all the features from the first state description, as well as angular momentum values for the torso, each arm and each leg, the foot orientations, and a measure of the hip abduction for each leg, giving a total of 39 features. For both representations, scaling factors were applied so that the range of values taken by each features was close to [-1, 1].

Models were trained with both a standard SVM approach, using the GPU version of LIBSVM version 3.17, and the local deep kernel learning (LDKL) variant of SVM, using LDKL version 1.1. The accuracy results and average prediction times for the two appraoches are shown in Table 6.6.1 and Table 6.6.2. Note that the prediction times for standard SVM are recorded in milliseconds, while the LDKL results are in microseconds, 3 orders of magnitude faster. The accuracy values obtained with LDKL are also higher. This differs from the results reported by Jose et al. (2013), where LDKL was typically found to have slightly lower accuracy, although it may be possible to improve the SVM accuracy results with additional tuning of the kernel parameters. Furthermore, the models trained using LDKL required only a few kilobytes of memory to store, while the SVM models used several megabytes of memory. This could be an important concern for making predictions online in real-time applications, where memory bandwidth is often a major constraint on performance. For this prediction problem, clearly LDKL has many advantages over the standard SVM approach. One disadvantage is that the LDKL implementation is free only for non-commercial use, while freely usable software such as LIBSVM is available for the standard SVM method. The higher dimensional feature space produced more accurate predictions, with a relatively small increase in prediction cost. The accuracy values are also high enough for the predictions to be expected to be useful. The video accompanying this section shows results using LDKL for a series of disturbances, with character states predicted to lead to failure shaded in red.

Features	Kernel	Degree	Training Time / s	Accuracy / %	Time per Prediction / ms
39	RBF	3	22.9	97.0	0.322
39	poly	3	50.8	95.3	0.822
13	RBF	3	13.4	96.7	0.177
13	poly	3	14.0	95.0	0.202

Table 6.6.1: LIBSVM Results

Features	Tree Depth	Training Time / s	Accuracy / %	Time per Prediction / μs
39	3	3.6	98.9	0.231
39	4	6.8	98.9	0.323
13	3	3.0	98.5	0.184
13	4	5.4	95.6	0.224

Table 6.6.2: LDKL Results

With LDKL prediction costs of the order of microseconds, it would be feasible to evaluate many controllers in a single time step, allowing for improved robustness strategies and transitions between dense networks of controllers to be implemented in a practical real-time application.

6.7 Summary

This chapter described the development of a feedback policy to stabilize the cyclic open loop control controller from Chapter 5. The feedback controller was based on existing work, but as with other aspects of the control problem, adaptations were required in order to achieve stability with large simulation time steps. In particular, the implementation here only produced successful controllers using dense feedback, rather than reduced order policies, and with the introduction of phase-dependence.

The successful feedback controller was retrained in a test environment including random external pushes, and a controller was generated which was able to recover from pushes of several hundred Newtons applied for 0.1s. The robustness to disturbances was in a similar range to most previous results, many of which use much smaller simulation time steps.

This chapter also included the use of machine learning techniques to predict whether a given controller will remain stable based on the current state of the character, which has many useful applications such as controlling transitions between controllers and improving robustness. Using a recent development in support vector machines, prediction costs of the order of microseconds, with an accuracy of 98.9%, were obtained. This improvement in prediction cost by several orders of magnitude compared to previous results could allow high level control strategies which involve evaluating many low level controllers in real-time to be explored.

Chapter 7

Conclusion

7.1 Summary

This project examined the applicability of physics-based locomotion control to simulated characters using 60 time steps per second, a simulation condition suitable for satisfying the performance constraints of typical games. A review was presented of the large body of existing work, which includes significant advances but typically uses much smaller simulation time steps, often around 2000 time steps per second. The difficulties of achieving stable control with larger time steps were illustrated with an attempt to implement a handcrafted controller similar to the well-known SIMBICON approach, at 60 time steps per second (SIMBICON itself was found to fail when the simulation rate was modified below 750 time steps per second). Despite numerous attempts to improve tracking quality and stability, control was severely hampered by tracking latency and failed after 1 or 2 motion cycles at most. This suggested that a more sophisticated approach was required to deal with large simulation steps, and the remainder of the project considered three major areas contributing to such an approach — trajectory optimization, to improve the physical plausibity of reference motions, sampling-based control, to generate open loop control policies, and learning feedback policies, to compensate for departures from desired motion.

The first area, trajectory optimization, was based on previous work which demonstrated that ground contacts could be incorporated into a direct optimization approach, but was limited to planar examples. Here, the method was extended to three dimensional problems. The inclusion of ground contacts in the optimization approach also introduces complementarity constraints into the optimization problem, which make solving difficult. Here, an exact penalty method was shown to give improved solver robustness and performance compared to the previously used

constraint relaxation method. Using the exact penalty method, optimized trajectories were successfully generated for walking, jogging and running motions, each taking around 25 minutes with a single-threaded solver. Trajectory optimization was also used to make significant alterations to characteristics such as timing, stride length and heading direction, without compromising physical plausibility, and to generate short transitions between existing motions. The trajectory optimization method was implemented using IPOPT, an open source solver, which has benefits for commercial use compared to previous work with proprietary solvers.

The second area of work used an existing sampling-based method to generate open loop controllers which reproduce reference motions, with adaptations to deal with larger simulation time steps. The simulation environment was also changed to PhysX, which is more representative of the kind of physics engines which would be common in games, but required some care to ensure deterministic behaviour, a requirement of the sampling technique. It was found that determinism in PhysX depended on various factors, including the method used to restore the character's state, the solver iteration count, internally cached contact information and the size of the slightly expanded region surrounding the feet which could trigger contacts. With suitable settings, it was possible to get highly consistent behaviour, but divergence still occurred after a few hundred time steps.

With large time steps, the sampling-based method was found to be sensitive to the physical plausibility of the reference motions used. Using trajectory optimization, it was possible to generate visually similar motions which resulted in successful open loop control policies at 60 time steps per second.

Automatic tuning of the dimensions of the sampling space was introduced, and it was found that the quality of the sampling output was sensitive to the sampling dimensions, and less dependent on the number of samples used.

A technique was also presented to generate cyclic control policies from the sample-based trajectories, which are inherently non-cyclic. The method involved searching for stretches of control which had small cyclic discontinuities, and creating a bridging segment using an evolutionary algorithm. Using this approach it was possible to obtain cyclic control policies which avoid large discontinuities between cycles, but with reduced stability compared to non-cyclic policies, leading to failure after around 3 motion cycles.

In any case, open loop control is useless on its own for interactive applications where it is necessary to deal with disturbances. Therefore, the final area of work involved generating feedback policies using an evolutionary approach, in order to correct for deviations from desired behaviour. With large time steps, it was found that it

was necessary to use a dense feedback structure, and to introduce phase-dependent feedback. With these conditions, the feedback controller was successful in stabilizing a walking motion at 60 time steps per second. When trained in an environment with random external pushes, the controller was able to recover from disturbances of several hundred Newtons, applied for 0.1s. The robustness to disturances was within the same range as previous work, much of which relies on substantially smaller time steps.

Using a recent machine learning method based on support vector machines, it was found that it was possible to predict whether the controller could recover from a given disturbed state, with high accuracy (98.9% for the example here) and with prediction times of the order of microseconds, orders of magnitude faster than previous work. Such predictions could be useful for numerous reasons, including gracefully handling failure, increasing robustness through composition of multiple controllers, managing transitions between different motions, and high level control strategies, and the prediction performance means that these techniques would be viable in games.

Together, these three areas of work allowed a walking motion to be successfully reproduced in simulation at 60 time steps per second. The total cost of control and simulation was 0.62ms per time step, giving a total processor utilization of around 4%, at least an order of magnitude lower than almost all previous work, and low enough to feasibly be used in applications with demanding performance requirements such as games. Furthermore, the availability of accurate controller stability prediction, also fast enough to be used in games, opens up exploration of various high level control strategies in real-time.

7.2 Discussion, Limitations and Future Work

There are several potential improvements to the trajectory optimization implementation to be considered. Some trial and error is currently used in selecting the penalty weight for the exact penalty method and it may be worthwhile to explore more systematic methods of selecting a weight value. For example, Benson et al. (2006) suggest a method based on estimates of the Lagrange multipliers at the initialization point. At present, multiple solver runs with successively increasing penalty weights are used. It may be possible to avoid the need for multiple solver runs by updating the penalty weight during a single solve, but care might be required to avoid interfering with the criterion used by IPOPT to accept steps based on the history of previous iterations. It may also be possible to improve the performance of the optimization by further exploring problem scaling, using a pre-solve step to get a better initialization point, or by using a multithreaded or GPU based linear

solver.

For motions modified using TO which are very far from the original reference, it can currently be difficult to avoid unnatural behaviour, and it is sometimes necessary to add objective terms to avoid implausible joint configurations or self-collision. Adding physiologically based joint limits and torque limits to the problem formulation may improve the naturalness of synthesized motions, and reduce the need for ad hoc objective terms.

TO is used to generate motion graphs in work by Ren et al. (2010) and Wampler et al. (2013), but the resulting outputs are purely kinematic. The results of Chapter 4 showed that with the TO implementation here it was possible to maintain physical plausibility while significantly altering reference motions. Therefore, an interesting future exploration would be to generate a densely connected graph of physically correct motions, based on a sparse initial set of references. The graph could then be used either with a low-dimensional model in order to create a pseudo-physical controller, or as the target behaviour for simulated control.

Although the technique used for making cyclic open loop control policies was successful, it was not wholly satisfactory. Problems with the current approach include limiting stability to only a few cycles, and significantly increasing the deviation from the target motion near the cycling boundary due to the difficulty of satisfying the looping restriction. In future, it would be desirable to develop a more effective approach, perhaps by using evolutionary optimization over the whole interval.

The samples generated during the open loop control policy determination and automatically tuned sampling window dimensions may contain a lot of information about a stable limit cycle and its basin of attraction. It could be worthwhile to explore strategies based on exploiting this information to generate a more stable limit cycle and feedback policy.

Feedback control has been successfully demonstrated at 60 time steps per second, but so far only for a walking motion. In order to examine its generality it will be important to test other motions, including more agile behaviours such as tight turning or running. Running motions in particular may be challenging due to the flight phases, although the successful application of TO to a high time step jogging motion is encouraging. If feedback control is found to be successful for other motions as well, then more flexible and high level control could be explored, either using a motion graph approach or controller parameterization.

The obtained walking feedback controller has somewhat jerky foot plants. This may be ameliorated in future by modifying the objective function or by improving the cyclic open loop control policy generation as described above.

The feedback controller is currently quite time consuming to train, requiring several hours per motion. Although it is amenable to parallelization across multiple motions, if the controller is to be extended to dense motion graphs it would be beneficial to explore ways of accelerating this step, such as using GPU based physics simulation.

As decribed in Section 6.6, there are numerous potential applications of fast, accurate controller stability prediction. One idea would be to train additional feedback controllers to increase robustness. For example, in Chapter 6, a single feedback controller was trained to deal with external forces from all directions. A controller which only has to cope with a narrower range of directions would be expected to have higher robustness for its area of applicability, and it should be possible to train several controllers for more specific directions, and automatically select between them using prediction. Similarly, it would be interesting to see if robustness may be improved by using prediction to allow the controller to select a different point in the open loop control sequence following a disturbance. Finally, prediction may be used to arbitrate high level control strategies. For example, if controllers have been trained for level ground and various slopes, the most appropriate controller may be selected based on prediction when the character encounters a change in slope. Similarly, prediction could be used to determine the most suitable point to transition to a different motion, for instance when a change in speed or heading is desired.

Therefore, the successful high time step feedback controller and efficient prediction form a promising basis for much interesting future exploration.

References

Abe, Y., da Silva, M. & Popović, J. (2007), Multiobjective control with frictional contacts, *in* 'Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation', Eurographics Association, pp. 249–258.

Abe, Y., Liu, C. & Popović, Z. (2006), 'Momentum-based parameterization of dynamic character motion', *Graphical models* **68**(2), 194–211.

Abend, W., Bizzi, E. & Morasso, P. (1982), 'Human arm trajectory formation', *Brain* **105**(2), 331.

Al Borno, M., de Lasa, M. & Hertzmann, A. (2013), 'Trajectory optimization for full-body movements with complex contacts', *Visualization and Computer Graphics, IEEE Transactions on* **19**(8), 1405–1414.

Al Borno, M., Fiume, E., Hertzmann, A. & de Lasa, M. (2014), Feedback control for rotational movements in feature space, *in* 'Computer Graphics Forum', Vol. 33, Wiley Online Library, pp. 225–233.

Alexander, R. (2001), 'Design by numbers', *Nature* **412**(6847), 591.

Allen, B. F. & Faloutsos, P. (2012), Misconceptions of PD control in animation, *in* 'Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation', Eurographics Association, pp. 231–234.

Allen, B. F., Neff, M. & Faloutsos, P. (2011), Analytic proportional-derivative control for precise and compliant motion, *in* 'Robotics and Automation (ICRA), 2011 IEEE International Conference on', IEEE, pp. 6039–6044.

Allen, B. & Faloutsos, P. (2009), 'Evolved controllers for simulated locomotion', *Motion in Games* pp. 219–230.

Amestoy, P. R., Duff, I. S., L'Excellent, J.-Y. & Koster, J. (2001), 'A fully asynchronous multifrontal solver using distributed dynamic scheduling', *SIAM Journal on Matrix Analysis and Applications* **23**(1), 15–41.

Anderson, F. & Pandy, M. (2001), 'Dynamic optimization of human walking', *Journal of Biomechanical Engineering* **123**, 381.

Andersson, J. (2013), A general-purpose software framework for dynamic optimization, PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Anitescu, M. (2005), 'On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints', *SIAM Journal on Optimization* **15**(4), 1203–1236.

Arutyunyan, G., Gurfinkel, V. & Mirskii, M. (1968), 'Investigation of aiming at a target', *Biophysics* **13**, 536–538.

Arutyunyan, G., Gurfinkel, V. & Mirskii, M. (1969), 'Organization of movements on execution by man of an exact postural task', *Biophysics* **14**(6), 1103–1107.

Baraff, D. (1996), Linear-time dynamics using lagrange multipliers, *in* 'Proceedings of the 23rd annual conference on Computer graphics and interactive techniques', ACM, pp. 137–146.

Bender, J., Erleben, K. & Trinkle, J. (2014), Interactive simulation of rigid body dynamics in computer graphics, *in* 'Computer Graphics Forum', Vol. 33, Wiley Online Library, pp. 246–270.

Benson, H. Y., Sen, A., Shanno, D. F. & Vanderbei, R. J. (2006), 'Interior-point algorithms, penalty methods and equilibrium problems', *Computational Optimization and Applications* **34**(2), 155–182.

Benson, H. Y., Shanno, D. F. & Vanderbei, R. J. (2003), A comparative study of large-scale nonlinear optimization algorithms, *in* 'High performance algorithms and software for nonlinear optimization', Springer, pp. 95–127.

Bent, L., Inglis, J. & McFadyen, B. (2004), 'When is vestibular information important during walking?', *Journal of neurophysiology* **92**(3), 1269–1275.

Betts, J. T. & Gablonsky, J. M. (2002), A comparison of interior point and SQP methods on optimal control problems, Technical Report M&CT-TECH-02-004, Mathematics and Computing Technology, Phantom Works, The Boeing Company, Seattle.

Bizzi, E., Hogan, N., Mussa-Ivaldi, F. A. & Giszter, S. (1992), 'Does the nervous system use equilibrium-point control to guide single and multiple joint movements?', *Behavioral and brain sciences* **15**(04), 603–613.

Blickhan, R. & Full, R. (1993), 'Similarity in multilegged locomotion: Bouncing like a monopode', *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology* **173**(5), 509–517.

Bodenheimer, B., Shleyfman, A. & Hodgins, J. (1999), The effects of noise on the perception of animated human running, *in* N. Magnenat-Thalmann & D. Thalmann, eds, 'Computer Animation and Simulation 99', Eurographics, Springer Vienna, pp. 53–63.

Boeing, A. & Bräunl, T. (2007), Evaluation of real-time physics simulation systems, *in* 'Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia', ACM, pp. 281–288.

Bongers, R., Fernandez, L. & Bootsma, R. (2009), 'Linear and logarithmic speed–accuracy trade-offs in reciprocal aiming result from task-specific parameterization of an invariant underlying dynamics.', *Journal of Experimental Psychology: Human Perception and Performance* **35**(5), 1443.

Brotman, L. & Netravali, A. (1988), Motion interpolation by optimal control, *in* 'ACM SIGGRAPH Computer Graphics', Vol. 22, ACM, pp. 309–315.

Brown, D. F., Macchietto, A., Yin, K. & Zordan, V. (2013), Control of rotational dynamics for ground behaviors, *in* 'Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation', ACM, pp. 55–61.

Cappellini, G., Ivanenko, Y., Poppele, R. & Lacquaniti, F. (2006), 'Motor patterns in human walking and running', *Journal of neurophysiology* **95**(6), 3426–3437.

Carvalho, S., Boulic, R. & Thalmann, D. (2007), 'Interactive low-dimensional human motion synthesis by combining motion models and PIK', *Computer Animation and Virtual Worlds* **18**(4-5), 493–503.

Cohen, M. (1992), Interactive spacetime control for animation, *in* 'ACM SIGGRAPH Computer Graphics', Vol. 26, ACM, pp. 293–302.

Collins, S., Ruina, A., Tedrake, R. & Wisse, M. (2005), 'Efficient bipedal robots based on passive-dynamic walkers', *Science* **307**(5712), 1082–1085.

Coros, S., Beaudoin, P. & van de Panne, M. (2010), 'Generalized biped walking control', *ACM Transactions on Graphics (TOG)* **29**(4), 130.

Coros, S., Beaudoin, P., Yin, K. & van de Panne, M. (2008), Synthesis of constrained walking skills, *in* 'ACM Transactions on Graphics (TOG)', Vol. 27, ACM, p. 113.

Cutting, J. & Kozlowski, L. (1977), 'Recognizing friends by their walk: Gait perception without familiarity cues', *Bulletin of the Psychonomic Society* **9**(5), 353–356.

da Silva, M., Abe, Y. & Popović, J. (2008*a*), Interactive simulation of stylized human locomotion, *in* 'ACM SIGGRAPH 2008 papers', ACM, pp. 1–10.

da Silva, M., Abe, Y. & Popović, J. (2008*b*), Simulation of human motion data using short-horizon model-predictive control, *in* 'Computer Graphics Forum', Vol. 27, Wiley Online Library, pp. 371–380.

da Silva, M., Durand, F. & Popović, J. (2009), 'Linear Bellman combination for control of character animation', *ACM Transactions on Graphics (TOG)* **28**(3), 1–10.

de Lasa, M., Mordatch, I. & Hertzmann, A. (2010), 'Feature-based locomotion controllers', *ACM Transactions on Graphics (TOG)* **29**(4), 131.

Ding, K., Liu, L., van de Panne, M. & Yin, K. (2012), Learning reduced-order feedback policies for motion skills, Technical report, Tech. Rep. TR-2012-06, University of British Columbia.

Egges, A., Molet, T. & Magnenat-Thalmann, N. (2004), Personalised real-time idle motion synthesis, *in* 'Proceedings of the Computer Graphics and Applications, 12th Pacific Conference', PG '04, IEEE Computer Society, Washington, DC, USA, pp. 121–130.

Erez, T., Tassa, Y. & Todorov, E. (2015), Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX, *in* 'IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015', pp. 4397–4404.

Faloutsos, P. (2002), Composable controllers for physics-based character animation, PhD thesis, University of Toronto.

Faloutsos, P., van de Panne, M. & Terzopoulos, D. (2001), Composable controllers for physics-based character animation, *in* 'Proceedings of the 28th annual conference on Computer graphics and interactive techniques', ACM, pp. 251–260.

Fang, A. & Pollard, N. (2003), 'Efficient synthesis of physically valid human motion', *ACM Transactions on Graphics (TOG)* **22**(3), 417–426.

Farley, C. & Morgenroth, D. (1999), 'Leg stiffness primarily depends on ankle stiffness during human hopping', *Journal of Biomechanics* **32**(3), 267–273.

Featherstone, R. (2014), *Rigid body dynamics algorithms*, Springer.

Franke, R. & Arnold, E. (1999), The solver Omuses/HQP for structured largescale constrained optimization: algorithm, implementation and example application, *in* 'Sixth SIAM conference on optimization'.

Fukushima, M., Luo, Z.-Q. & Pang, J.-S. (1998), 'A globally convergent sequential quadratic programming algorithm for mathematical programs with linear complementarity constraints', *Computational Optimization and Applications* **10**(1), 5–34.

Geijtenbeek, T., Pronost, N., Egges, A. & Overmars, M. H. (2011), 'Interactive character animation using simulated physics', *Eurographics-state of the art reports* **2**.

Geijtenbeek, T., van de Panne, M. & van der Stappen, A. F. (2013), 'Flexible muscle-based locomotion for bipedal creatures', *ACM Transactions on Graphics (TOG)* **32**(6), 206.

Geyer, H. & Herr, H. (2010), 'A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities', *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* **18**(3), 263–273.

Geyer, H., Seyfarth, A. & Blickhan, R. (2003), 'Positive force feedback in bouncing gaits?', *Proceedings of the Royal Society of London. Series B: Biological Sciences* **270**(1529), 2173–2183.

Geyer, H., Seyfarth, A. & Blickhan, R. (2006), 'Compliant leg behaviour explains basic dynamics of walking and running', *Proceedings of the Royal Society B: Biological Sciences* **273**(1603), 2861–2867.

Gibet, S., Kamp, J. & Poirier, F. (2004), 'Gesture analysis: Invariant laws in movement', *Gesture-Based Communication in Human-Computer Interaction* pp. 451–452.

Gilden, D. & Proffitt, D. (1989), 'Understanding collision dynamics.', *Journal of Experimental Psychology: Human Perception and Performance* **15**(2), 372.

Giovanni, S. & Yin, K. (2011), LocoTest: deploying and evaluating physics-based locomotion on multiple simulation platforms, *in* 'Motion in Games', Springer, pp. 227–241.

Gould, N. I., Scott, J. A. & Hu, Y. (2007), 'A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations', *ACM Transactions on Mathematical Software (TOMS)* **33**(2), 10.

Han, D., Noh, J., Jin, X., S Shin, J. & Y Shin, S. (2014), On-line real-time physics-based predictive motion control with balance recovery, *in* 'Computer Graphics Forum', Vol. 33, Wiley Online Library, pp. 245–254.

Hansen, N. (2006), 'The CMA evolution strategy: a comparing review', *Towards a new evolutionary computation* pp. 75–102.

Harris, C. & Wolpert, D. (1998), 'Signal-dependent noise determines motor planning', *Nature* **394**(6695), 780–784.

Herr, H. & Popovic, M. (2008), 'Angular momentum in human walking', *Journal of Experimental Biology* **211**(4), 467.

Hodgins, J., Jörg, S., O'Sullivan, C., Park, S. & Mahler, M. (2010), 'The saliency of anomalies in animated human characters', *ACM Transactions on Applied Perception (TAP)* **7**(4), 1–14.

Hodgins, J., O'Brien, J. & Tumblin, J. (1998), 'Perception of human motion with different geometric models', *Visualization and Computer Graphics, IEEE Transactions on* **4**(4), 307–316.

Hodgins, J., Wooten, W., Brogan, D. & O'Brien, J. (1995), Animating human athletics, *in* 'Proceedings of the 22nd annual conference on Computer graphics and interactive techniques', ACM, pp. 71–78.

Hollands, M. & Marple-Horvat, D. (1996), 'Visually guided stepping under conditions of step cycle-related denial of visual information', *Experimental brain research* **109**(2), 343–356.

Ivanenko, Y., Poppele, R. & Lacquaniti, F. (2004), 'Five basic muscle activation patterns account for muscle activity during human locomotion', *The Journal of physiology* **556**(1), 267–282.

Jain, S. & Liu, C. K. (2011*a*), Controlling physics-based characters using soft contacts, *in* 'ACM Transactions on Graphics (TOG)', Vol. 30, ACM, p. 163.

Jain, S. & Liu, C. K. (2011*b*), 'Modal-space control for articulated characters', *ACM Transactions on Graphics (TOG)* **30**(5), 118.

Jain, S., Ye, Y. & Liu, C. (2009), 'Optimization-based interactive motion synthesis', *ACM Transactions on Graphics (TOG)* **28**(1), 1–12.

Jenkins, O. C. & Mataric, M. J. (2003), Automated derivation of behavior vocabularies for autonomous humanoid motion, *in* 'Proceedings of the second international joint conference on Autonomous agents and multiagent systems', ACM, pp. 225–232.

Johansson, G. (1973), 'Visual perception of biological motion and a model for its analysis', *Attention, Perception, & Psychophysics* **14**(2), 201–211.

Johansson, G. (1976), 'Spatio-temporal differentiation and integration in visual motion perception', *Psychological Research* **38**(4), 379–393.

Jose, C., Goyal, P., Aggrwal, P. & Varma, M. (2013), Local deep kernel learning for efficient non-linear SVM prediction, *in* 'Proceedings of the 30th International Conference on Machine Learning (ICML-13)', pp. 486–494.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K. & Hirukawa, H. (2003), Biped walking pattern generation by using preview control of zero-moment point, *in* 'Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on', Vol. 2, Ieee, pp. 1620–1626.

Karypis, G. & Kumar, V. (1998), 'A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices', *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN* .

Kawato, M. (1999), 'Internal models for motor control and trajectory planning', *Current opinion in neurobiology* **9**(6), 718–727.

Komura, T., Leung, H. & Kuffner, J. (2004), Animating reactive motions for biped locomotion, *in* 'Proceedings of the ACM symposium on Virtual reality software and technology', ACM, pp. 32–40.

Kry, P. G., Revéret, L., Faure, F. & Cani, M.-P. (2009), Modal locomotion: Animating virtual characters with natural vibrations, *in* 'Computer Graphics Forum', Vol. 28, Wiley Online Library, pp. 289–298.

Kubow, T., Schmitt, J., Holmes, P. & Koditschek, D. (2002), 'Quantifying dynamic stability and maneuverability in legged locomotion', *Integrative and Comparative Biology* **42**(1), 149.

Kwon, T. & Hodgins, J. (2010), Control systems for human running using an inverted pendulum model and a reference motion capture sequence, *in* 'Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation', Eurographics Association, pp. 129–138.

Lackner, J. & Dizio, P. (1994), 'Rapid adaptation to Coriolis force perturbations of arm trajectory', *Journal of neurophysiology* **72**(1), 299–313.

Laszlo, J., van de Panne, M. & Fiume, E. (1996), Limit cycle control and its application to the animation of balancing and walking, *in* 'Proceedings of the 23rd annual conference on Computer graphics and interactive techniques', ACM, pp. 155–162.

Lee, S. & Goswami, A. (2010), Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground, *in* 'Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on', IEEE, pp. 3157–3162.

Lee, Y., Kim, S. & Lee, J. (2010), 'Data-driven biped control', *ACM Transactions on Graphics (TOG)* **29**(4), 129.

Liu, C., Hertzmann, A. & Popović, Z. (2005), 'Learning physics-based motion style with nonlinear inverse optimization', *ACM Transactions on Graphics (TOG)* **24**(3), 1071–1081.

Liu, C. K., Hertzmann, A. & Popović, Z. (2006), Composition of complex optimal multi-character motions, *in* 'Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation', Eurographics Association, pp. 215–222.

Liu, C. K. & Popović, Z. (2002), Synthesis of complex dynamic character motion from simple animations, *in* 'ACM Transactions on Graphics (TOG)', Vol. 21, ACM, pp. 408–416.

Liu, F., Southern, R., Guo, S., Yang, X. & Zhang, J. J. (2013), 'Motion adaptation with motor invariant theory', *Cybernetics, IEEE Transactions on* **43**(3), 1131–1145.

Liu, L., Yin, K., van de Panne, M. & Guo, B. (2012), 'Terrain runner: control, parameterization, composition, and planning for highly dynamic motions.', *ACM Trans. Graph.* **31**(6), 154.

Liu, L., Yin, K., van de Panne, M., Shao, T. & Xu, W. (2010), 'Sampling-based contact-rich motion control', *ACM Transactions on Graphics (TOG)* **29**(4), 128.

Liu, Z. & Cohen, M. (1995), Keyframe motion optimization by relaxing speed and timing, *in* '6th Eurographics Workshop on Animation and Simulation', pp. 144–153.

Liu, Z., Gortler, S. & Cohen, M. (1994), Hierarchical spacetime control, *in* 'Proceedings of the 21st annual conference on Computer graphics and interactive techniques', ACM, pp. 35–42.

Luksch, T. et al. (2010), Human-like control of dynamically walking bipedal robots, PhD thesis, Technischen Universität Kaiserslautern.

Macchietto, A., Zordan, V. & Shelton, C. (2009), 'Momentum control for balance', *ACM Transactions on Graphics (TOG)* **28**(3), 80.

Matsuoka, K. (1985), 'Sustained oscillations generated by mutually inhibiting neurons with adaptation', *Biological cybernetics* **52**(6), 367–376.

McGeer, T. (1990), 'Passive dynamic walking', *The International Journal of Robotics Research* **9**(2), 62.

Mirtich, B. (1996), Impulse-based dynamic simulation of rigid body systems, PhD thesis, University of California, Berkeley.

Mordatch, I., de Lasa, M. & Hertzmann, A. (2010), 'Robust physics-based locomotion using low-dimensional planning', *ACM Transactions on Graphics (TOG)* **29**(4), 71.

Mordatch, I., Todorov, E. & Popović, Z. (2012), 'Discovery of complex behaviors through contact-invariant optimization', *ACM Transactions on Graphics (TOG)* **31**(4), 43.

Mordatch, I., Wang, J. M., Todorov, E. & Koltun, V. (2013), 'Animating human lower limbs using contact-invariant optimization', *ACM Transactions on Graphics (TOG)* **32**(6), 203.

Mori, M. (1970), 'The uncanny valley', *Energy* **7**(4), 33–35.

Muico, U., Lee, Y., Popović, J. & Popović, Z. (2009), 'Contact-aware nonlinear control of dynamic characters', *ACM Transactions on Graphics (TOG)* **28**(3), 1–9.

Muico, U., Popović, J. & Popović, Z. (2011), 'Composite control of physically simulated characters', *ACM Transactions on Graphics (TOG)* **30**(3), 16.

Neff, M. & Fiume, E. (2002), Modeling tension and relaxation for computer animation, *in* 'Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation', ACM, pp. 81–88.

Ngo, J. & Marks, J. (1993), Spacetime constraints revisited, *in* 'Proceedings of the 20th annual conference on Computer graphics and interactive techniques', ACM, pp. 343–350.

Novacheck, T. (1998), 'The biomechanics of running', *Gait & Posture* **7**(1), 77–95.

Oesker, M., Hecht, H. & Jung, B. (2000), 'Psychological evidence for unconscious processing of detail in real-time animation of multiple characters', *The Journal of Visualization and Computer Animation* **11**(2), 105–112.

O'Sullivan, C. & Dingliana, J. (2001), 'Collisions and perception', *ACM Transactions on Graphics (TOG)* **20**(3), 151–168.

O'Sullivan, C., Dingliana, J., Giang, T. & Kaiser, M. (2003), Evaluating the visual fidelity of physically based animations, *in* 'ACM Transactions on Graphics (TOG)', Vol. 22, ACM, pp. 527–536.

Perlin, K. (1995), 'Real time responsive animation with personality', *Visualization and Computer Graphics, IEEE Transactions on* **1**(1), 5–15.

Pollard, N. & Reitsma, P. (2001), Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model, *in* 'Yale Workshop on Adaptive and Learning Systems'.

Popović, Z. & Witkin, A. (1999), Physically based motion transformation, *in* 'Proceedings of the 26th annual conference on Computer graphics and interactive techniques', ACM Press/Addison-Wesley Publishing Co., pp. 11–20.

Posa, M., Cantu, C. & Tedrake, R. (2014), 'A direct method for trajectory optimization of rigid bodies through contact', *The International Journal of Robotics Research* **33**(1), 69–81.

Pozzo, T., Berthoz, A. & Lefort, L. (1990), 'Head stabilization during various locomotor tasks in humans', *Experimental Brain Research* **82**(1), 97–106.

Pratt, J., Carff, J., Drakunov, S. & Goswami, A. (2006), Capture point: A step toward humanoid push recovery, *in* 'Humanoid Robots, 2006 6th IEEE-RAS International Conference on', IEEE, pp. 200–207.

Pratt, J. E. & Krupp, B. T. (2004), Series elastic actuators for legged robots, *in* 'Defense and Security', International Society for Optics and Photonics, pp. 135–144.

Pražák, M., McDonnell, R. & O'Sullivan, C. (2010), Perceptual evaluation of human animation timewarping, *in* 'ACM SIGGRAPH ASIA 2010 Sketches', ACM, p. 30.

Raibert, M. & Hodgins, J. (1991), Animation of dynamic legged locomotion, *in* 'ACM SIGGRAPH Computer Graphics', Vol. 25, ACM, pp. 349–358.

Reil, T. & Husbands, P. (2002), 'Evolution of central pattern generators for bipedal walking in a real-time physics environment', *Evolutionary Computation, IEEE Transactions on* **6**(2), 159–168.

Reil, T. & Massey, C. (2001), 'Biologically inspired control of physically simulated bipeds', *Theory in Biosciences* **120**(3), 327–339.

Reitsma, P. & Pollard, N. (2003), Perceptual metrics for character animation: sensitivity to errors in ballistic motion, *in* 'ACM SIGGRAPH 2003 Papers', ACM, pp. 537–542.

Ren, C., Zhao, L. & Safonova, A. (2010), Human motion synthesis with optimization-based graphs, *in* 'Computer Graphics Forum', Vol. 29, Wiley Online Library, pp. 545–554.

Ren, L., Patrick, A., Efros, A., Hodgins, J. & Rehg, J. (2005), 'A data-driven approach to quantifying natural human motion', *ACM Transactions on Graphics (TOG)* **24**(3), 1090–1097.

Safonova, A. & Hodgins, J. (2005), Analyzing the physical correctness of interpolated human motion, *in* 'Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation', ACM, pp. 171–180.

Safonova, A., Hodgins, J. & Pollard, N. (2004), Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces, *in* 'ACM SIGGRAPH 2004 Papers', ACM, pp. 514–521.

Shadmehr, R. & Mussa-Ivaldi, F. (1994), 'Adaptive representation of dynamics during learning of a motor task', *The Journal of Neuroscience* **14**(5), 3208–3224.

Sharon, D. & van de Panne, M. (2005), Synthesis of controllers for stylized planar bipedal walking, *in* 'Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on', IEEE, pp. 2387–2392.

Shin, H., Kovar, L. & Gleicher, M. (2003), Physical touch-up of human motions, *in* 'Proceedings of the 11th Pacific Conference on Computer Graphics and Applications', Vol. 194.

Sims, K. (1994), Evolving virtual creatures, *in* 'Proceedings of the 21st annual conference on Computer graphics and interactive techniques', ACM, pp. 15–22.

Sok, K., Kim, M. & Lee, J. (2007), Simulating biped behaviors from human motion data, *in* 'ACM SIGGRAPH 2007 papers', ACM, pp. 107–es.

Stewart, D. E. & Trinkle, J. C. (1996), 'An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction', *International Journal for Numerical Methods in Engineering* **39**(15), 2673–2691.

Takahashi, C., Scheidt, R. & Reinkensmeyer, D. (2001), 'Impedance control and internal model formation when reaching in a randomly varying dynamical environment', *Journal of neurophysiology* **86**(2), 1047.

Tassa, Y., Erez, T. & Todorov, E. (2012), Synthesis and stabilization of complex behaviors through online trajectory optimization, *in* 'Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on', IEEE, pp. 4906–4913.

Tedrake, R. (2004), Applied optimal control for dynamically stable legged locomotion, PhD thesis, Massachusetts Institute of Technology.

Todorov, E. (2004), 'Optimality principles in sensorimotor control (review)', *Nature neuroscience* **7**(9), 907.

Todorov, E. (2011), A convex, smooth and invertible contact model for trajectory optimization, *in* 'Robotics and Automation (ICRA), 2011 IEEE International Conference on', IEEE, pp. 1071–1076.

Troje, N. (2002), 'Decomposing biological motion: A framework for analysis and synthesis of human gait patterns', *Journal of vision* **2**(5).

Tsai, Y., Lin, W., Cheng, K., Lee, J. & Lee, T. (2009), 'Real-time physics-based 3d biped character animation using an inverted pendulum model', *IEEE transactions on visualization and computer graphics* pp. 325–337.

van de Panne, M. (2000), 'Control for simulated human and animal motion', *Annual Reviews in Control* **24**, 189–199.

van de Panne, M. & Fiume, E. (1993), Sensor-actuator networks, *in* 'Proceedings of the 20th annual conference on Computer graphics and interactive techniques', ACM, pp. 335–342.

van Soest, A. J. & Bobbert, M. F. (1993), 'The contribution of muscle properties in the control of explosive movements', *Biological cybernetics* **69**(3), 195–204.

Vukobratović, M. & Borovac, B. (2004), 'Zero-moment pointthirty five years of its life', *International Journal of Humanoid Robotics* **1**(01), 157–173.

Wächter, A. & Biegler, L. T. (2006), 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming', *Mathematical programming* **106**(1), 25–57.

Wagner, H. & Blickhan, R. (1999), 'Stabilizing function of skeletal muscles: an analytical investigation', *Journal of theoretical biology* **199**(2), 163–179.

Walther, A. & Griewank, A. (2012), 'Getting started with ADOL-C', *Combinatorial Scientific Computing* pp. 181–202.

Wampler, K., Popović, J. & Popović, Z. (2013), Animal locomotion controllers from scratch, *in* 'Computer Graphics Forum', Vol. 32, Wiley Online Library, pp. 153–162.

Wampler, K. & Popović, Z. (2009), Optimal gait and form for animal locomotion, *in* 'ACM Transactions on Graphics (TOG)', Vol. 28, ACM, p. 60.

Wampler, K., Popović, Z. & Popović, J. (2014), 'Generalizing locomotion style to new animals with inverse optimal regression', *ACM Trans. Graph.* **33**(4), 49:1–49:11.

Wang, J., Fleet, D. & Hertzmann, A. (2009), Optimizing walking controllers, *in* 'ACM Transactions on Graphics (TOG)', Vol. 28, ACM, p. 168.

Wang, J., Fleet, D. & Hertzmann, A. (2010), Optimizing walking controllers for uncertain inputs and environments, *in* 'ACM Transactions on Graphics (TOG)', Vol. 29, ACM, p. 73.

Wang, J., Hamner, S., Delp, S. & Koltun, V. (2012), 'Optimizing locomotion controllers using biologically-based actuators and objectives', *ACM Transactions on Graphics (TOG)* **31**(4), 25.

Witkin, A. & Kass, M. (1988), Spacetime constraints, *in* 'ACM Siggraph Computer Graphics', Vol. 22, ACM, pp. 159–168.

Wright, S. & Nocedal, J. (1999), *Numerical optimization*, Springer New York.

Wu, C. & Zordan, V. (2010), Goal-directed stepping with momentum control, *in* 'Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation', Eurographics Association, pp. 113–118.

Wu, J. & Popović, Z. (2010), 'Terrain-adaptive bipedal locomotion control', *ACM Transactions on Graphics (TOG)* **29**(4), 72.

Ye, Y. & Liu, C. K. (2010), 'Optimal feedback control for character animation using an abstract model', *ACM Transactions on Graphics (TOG)* **29**(4), 74.

Yin, K., Coros, S., Beaudoin, P. & van de Panne, M. (2008), Continuation methods for adapting simulated skills, *in* 'ACM SIGGRAPH 2008 papers', ACM, pp. 1–7.

Yin, K., Loken, K. & van de Panne, M. (2007), SIMBICON: simple biped locomotion control, *in* 'ACM SIGGRAPH 2007 papers', ACM, pp. 105–es.

Zajac, F. et al. (1989), 'Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control', *Critical reviews in biomedical engineering* **17**(4), 359.

Zehr, E. & Stein, R. (1999), 'What functions do reflexes serve during human locomotion?', *Progress in neurobiology* **58**(2), 185–205.

Zehr, E., Stein, R. & Komiyama, T. (1998), 'Function of sural nerve reflexes during human walking', *The Journal of Physiology* **507**(1), 305–314.

Zordan, V. & Hodgins, J. (2002), Motion capture-driven simulations that hit and react, *in* 'Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation', ACM, pp. 89–96.