Feature-Based Control of Physics-Based Character Animation

by

Martin de Lasa

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Feature-Based Control of
Physics-Based Character Animation

Martin de Lasa
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2011

Creating controllers for physics-based characters is a long-standing open problem in animation and robotics. Such controllers would have numerous applications while potentially yielding insight into human motion. Creating controllers remains difficult: current approaches are either constrained to track motion capture data, are not robust, or provide limited control over style.

This thesis presents an approach to control of physics-based characters based on high-level features of human movement, such as center-of-mass, angular momentum, and end-effector motion. Objective terms are used to control each feature, and are combined via optimization. We show how locomotion can be expressed in terms of a small number of features that control balance and end-effectors. This approach is used to build controllers for biped balancing, jumping, walking, and jogging.

These controllers provide numerous benefits: human-like qualities such as arm-swing, heel-off, and hip-shoulder counter-rotation emerge automatically during walking; controllers are robust to changes in body parameters; control parameters apply to intuitive properties; and controller may be mapped onto entirely new bipeds with different topology and mass distribution, without controller modifications. Transitions between multiple types of gaits, including walking, jumping, and jogging, emerge automatically. Controllers can traverse challenging terrain while following high-level user commands at interactive rates. This approach uses no motion capture or off-line optimization process.

Although we focus on the challenging case of bipedal locomotion, many other types of controllers stand to benefit from our approach.

Acknowledgements

Returning to graduate school after several years of working in the "real-world" was not easy for me. Were it not for the support and contributions of many people, this thesis would never have been completed.

First and foremost, I am indebted to my supervisor Aaron Hertzmann for his guidance, encouragement, collaboration, and financial support. Aaron's boundless energy, math chops, diverse research interests, and attention to detail have been a constant source of inspiration throughout my doctoral studies. The core ideas in this thesis are the result of countless meetings, thought experiments, and conversations with Aaron.

Many thanks are also due to my thesis committee members, David J. Fleet, Karan Singh, and to my external examiner Jovan Popović for their criticism and support throughout this process and for their specific help improving this document. I am especially grateful to David who always made himself available, providing a fresh perspective and insightful feedback on, too many, early paper and checkpoint document drafts.

Special thanks to my collaborator Igor Mordatch for all his hard work and dedication. His ideas and proposed solutions greatly contributed to the results described in this thesis. My graduate experience was enriched by working with and learning from Igor.

I would also like to thank the other members of the DGP lab for their friendship and support. Special thanks to Simon Breslav for patient understanding during many late night video editing sessions, to Mike Daum for being a sounding board and source of support, and to Jack Wang for many great exchanges and countless culinary adventures on Spadina. Thanks to the rest of the DGP crew for creating an open and fun work atmosphere, including: Anand Agarawala, Jacky Bibliowicz, Patrick Coleman, Sam Hasinoff, Christian Lessig, Noah Lockwood, Derek Nowrouzezahrai, Peter O'Donovan, Faisal Qureshi, Ryan Schmidt, Patricio Simari, and too many others to list individually.

To my adoptive office mates in GB418. Thanks for many memorable lunch breaks and showing me what life is like on the other side of St. George st.

I am very thankful for the financial support provided by the National Sciences and Engineering Research Council of Canada (NSERC) and the University of Toronto's Computer Science

Department, which allowed me to focus on the task of research.

This work would have never happened without the encouragement and support of my entire family. To my parents, Hugo and Graciela, thank you for your love, support, and reminding me to focus on the "big picture". To my parents-in-law, Marianna and Stefan, thank you for your unbounded generosity and willingness to assume parental responsibilities on a phone call's notice. To my children, Stefan and Amelia, thank you for bringing chaos to my world and for reminding me of the truly important things in life. Lastly to my wife, Magda, thank you for embarking and accompanying me along this journey; no words can express my gratitude.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> From the motor chauvinist's point of view the entire purpose of the human brain is to produce movement. Movement is the only way we have of interacting with the world. All communication, including speech, sign language, gestures and writing, is mediated via the motor system
>
> [Wolpert et al. 2001]

From an early age, we learn to use our bodies to grasp objects and locomote. Long before we are able to speak, we use movement to let others know how we feel, express intent, and infer the mood of those around us. For humans, movement is a non-verbal extension of our personality that allows us to communicate, interact with, and shape the world we live in.

In spite of our innate familiarity with human movement, we still have a poor understanding of human motor control and learning processes. This remains one of the grand challenges of science, with practical implications across numerous domains including entertainment, security, and medicine.

The study of human motion dates back to Aristotle's *De Motu Animalium* (On the Movement of Animals) that advocated qualitative study of motion through observation of experiments. To date, this has remained the predominant methodology used by biomechanists, with the last hundred years seeing large advances in the use, and development, of quantitative models. Using detailed *in vivo* measurements of muscle activations, joint angles, and ground contact forces, a number of biomechanical models have been developed to explain gross aspects of motion. These models have been shown to hold, with surprising similarity, across a wide variety of multi-limbed animals, including bipeds, quadrupeds, hexapeds, and polypeds [Full and Koditschek

1999]. Energy storage and recovery principles have been shown to extend to non-terrestrial activities, such as swimming and flight [Dickinson et al. 2000].

To reduce complexity and provide tractable representations for analysis and understanding, low-dimensional biomechanical models typically ignore "unused" limbs, such as the head, trunk, and non-contact limbs; non-essential joints, such as vertebrae and knees, are also omitted. The simplest model used to describe bipedal walking, the inverted pendulum, consists of a rigid link connecting a point mass at the center-of-mass (COM) to the center-of-pressure (COP). This provides an explanation of why the body speeds up or slows down, during the single-stance phase of walking, but leaves many aspects of whole-body motion undefined. For example, it provides no insight into particular selections of foot plants or mechanisms responsible for arm/leg synchronization. Additionally, since models are typically planar and reduce the body to an inertialess point mass, they are restricted to small motion regimes.

Most biomechanical models are *descriptive*. For given recordings of animal motion, they provide hypotheses explaining observed behavior. Robotics and physics-based computer animation require *generative* models, capable of synthesizing motion from descriptions of desired behavior.

Unlike models of passive phenomena (e.g., cloth, fire, and water), generative models of human motion require *control*. Control is required to specify internal, time-varying, forces acting at joints. For locomotion, control must meet several goals. At its most basic level, locomotion control should keep the character from falling over when operating in steady-state or handling external perturbations and modeling errors. Ideally, locomotion control should be capable of producing a variety of stylized gaits and transitions, of adapting to different types of terrain, and of generalizing across characters with different anthropometry. The nature of human dynamics makes these design goals very challenging.

Human dynamics are high-dimensional, highly nonlinear, underactuated, change discretely, and are naturally unstable[1]. Hence, mathematical tools from classical control theory are of limited use. Furthermore, since contact forces between the character and the environment are constrained to be repulsive (i.e., contact forces cannot pull the feet towards the ground), available control actions are severely limited. For example, when standing, ankle torques must be carefully chosen to avoid foot rotation about the heel or ball of the foot. When compensating for

[1]Human dynamics are analogous to a ball on top of a mountain. Without constant monitoring and corrective actions the ball will roll of the mountain (i.e., the person will fall over). Naturally stable systems (e.g., a ball in a valley) have dynamics that remain unchanged even without any corrective actions [Ringrose 1997].

external disturbances, control must decide when ankle torques are sufficient and when it is necessary to take a step. Depending on the nature of the disturbance, specific foot plants may be preferable. For rugged terrain, where there is little margin for error, this problem becomes more difficult since control must account for the future impact of actions on motion completion. A number of additional practical challenges arise when targeting robotics systems, including actuator and power limitations, inaccurate knowledge of system inertial and frictional properties, sensor measurement error, and limited knowledge of global system state.

Despite these difficulties, researchers have been able to develop an ever growing body of control examples by iterative experimentation, analysis, and refinement. For humanoid locomotion, this pragmatic approach has produced impressive results but has, to date, failed to yield a unifying method. Most developed techniques exploit particular system details (e.g., many synthesis techniques have targeted planar systems without upper body limbs), making it very difficult to apply control algorithms to new characters with large topological or anthropomorphic differences. Methods also tend to be gait specific; control generalizes poorly to new actions or environments. Hence, a general theory of locomotion control based on common high-level biomechanical principles, or building blocks, has remained elusive.

1.1 Approach

This thesis aims to develop a general set of tools to ease design of control for simulated physics-based characters. Instead of formulating control in terms of low-level commands, such as individual joint actions which can vary in number and representation between different character models, we seek to identify essential motion features and to formulate control in terms of these high-level quantities.

By using higher-level abstractions, our goal is to produce control that is more resilient to variations in character and environmental properties. We call this approach, *feature-based control*.

This thesis aims to answer three fundamental questions about feature-based control:

- What is a sufficient set of features needed to produce locomotion?
- How can individual features be regulated?
- How can multiple features be combined to design robust controllers?

In addition to answering these scientific questions, we describe and solve many practical engineering challenges that must be overcome in order to obtain a working feature-based control system.

The methods we describe provide numerous important and desirable properties. Generated feature-based control generalizes to characters of very different topology and mass properties. Additionally, many natural properties of locomotion, such as toe-off, arm swing, and hip sway, emerge automatically without being explicitly specified. Resulting control can navigate over challenging uneven terrain, with steps, inclines, stepping-stones, and gaps, while responding to interactive high-level user commands (e.g., step-length, step duration, COM apex height, heading, etc.). We demonstrate robustness to aggressive external disturbances, applied both in stance and flight.

Although we believe ideas in this thesis will benefit humanoid control work in robotics, and generalize to other types of creatures, such as quadrupeds, we limit the scope of our work to developing control for simulated biped characters. Focusing on simulated characters avoids hardware and sensing difficulties, allowing us to concentrate on developing the needed mathematical tools to achieve our goals. From a practical perspective, there are a huge number of applications, ranging from video games to security, that will benefit from the development of a unified approach to control development.

1.2 Thesis Organization and Contributions

A review of control techniques for physics-based motion synthesis is first presented (Chapter 2), followed by three separate methods for low-dimensional control of physics-based characters in Chapters 3–5. Chapter 7 concludes with a discussion of the limitations of our approach; we touch upon several interesting directions for future research. A brief description of key contributions follows.

Feature-Based Control. We begin by introducing feature-based control and defining needed objectives for control of abstract high-level quantities (Chapter 3). We describe two novel objectives for moving between distant feature targets and regulating whole-body angular momentum without center-of-pressure knowledge. To validate the proposed control formulation, we present three problems of increasing complexity, illustrating how acceleration, torque,

and bilateral force constraints can be incorporated in our formulation. All examples run in real-time, while requiring less tuning effort and providing a number of benefits over previous methods.

Feature-Based Locomotion Control. Building upon the results of Chapter 3, Chapter 4 presents locomotion control examples that use unilateral constraints to model ground contact forces and joint/torque limits. We show how to hand-design control for complex actions, such as walking and jumping, by using finite-state machines to coordinate time-varying actions. Since control includes a model of character dynamics, it tolerates large changes in anthropometry and character topology. Hence, control can be applied to different characters, with only minor parameter changes.

Robust Physics-Based Locomotion Using Low-Dimensional Planning. Chapter 5 describes a novel planner aimed at increasing the robustness and versatility of feature-based control. Instead of hand-designing task-specific state machines, as described in Chapter 4, optimal features commands are automatically selected using nonlinear optimization. To make this planning efficient, we propose a low-dimensional model, based on linearized Spring Loaded Inverted Pendulum equations-of-motion, along with a schedule that plans feature-space actions over the next two footsteps. To select optimal actions, we propose an objective that considers a number of stylistic motion characteristics, such as direction, step length, step duration, and COM height. In conjunction, these elements generate a continuum of robust behaviors over uneven terrain with aggressive variations.

Prioritized Optimization. Chapter 6 presents mathematical details of the prioritized optimization algorithms used throughout Chapters 3 and 4. In prioritized optimization a nested sequence of objectives is optimized so as not to conflict with higher-priority objectives. We derive two recursive algorithms for problems with equality and inequality constraints. These algorithms can be viewed as solutions to lexicographic optimization for the special case of quadratic objectives with linear constraints in free variables (i.e., quadratic programs). To the best of our knowledge, we are the first to design algorithms for this important case. Furthermore, we show that when all objectives are placed at the same priority level, our formulation reduces to a weighted multiobjective optimization. In the context of control, these algorithms can be used to enforce task-relevant acceleration, torque, and force constraints.

Contributions in the aforementioned chapters have been previously published in the ACM Transactions on Graphics (Proceedings of SIGGRAPH) [de Lasa et al. 2010; Mordatch et al. 2010] and the Proceedings of the International Conference on Intelligent Robots and Systems (IROS) [de Lasa and Hertzmann 2009]. We refer to accompanying videos containing important illustrations of results throughout the thesis. They can be found at:

Chapter 3 `http://www.dgp.toronto.edu/~mdelasa/iros09`

Chapter 4 `http://www.dgp.toronto.edu/~mdelasa/feature`

Chapter 5 `http://www.dgp.toronto.edu/~mdelasa/slip`

Chapter 2

Previous Work

Real-world motion is the result of muscles, gravity, and external forces. By modeling these elements, physics-based character animation holds the promise of generating nuanced and expressive motion, with human-like qualities. In recent years, great strides has been made towards this goal, with state-of-the-art systems far exceeding capabilities of just a decade ago.

In this section, we summarize common themes from the expansive body of work on locomotion research in animation, robotics, biology, neuroscience, and biomechanics. We seek to distill similarities, with the aim of informing the principal goal of this thesis: designing computational models of robust locomotion control for complex physically-simulated biped characters.

Because of the tremendous overlap in representations, constraints, and methods used throughout the literature, it is difficult to neatly classify previous work into distinct categories. To complicate matters, research goals vary widely. Biomechanists and integrative biologists seek insight into mechanisms used by real organisms, without concern for synthetic implementation; explaining observations is often enough. Neuroscientists focus on the brain and are concerned with computational and memory-based models of motor control. Roboticists focus on developing control for real systems, operating in real environments; motion quality is often of secondary importance. Animation researchers are concerned, above all else, with motion quality, "style", and end-user controllability; accurately mimicking biological mechanisms is not a primary goal. Standards for evaluating results reflect these differences, as well as domain-specific challenges.

2.1 Simplified Models

> A key challenge in the study of locomotion is to determine how each individual component within the locomotor system operates, while at the same time discovering how they function collectively as an integrated whole.
>
> [Dickinson et al. 2000]

Reasoning about whole-body motion of complex characters is difficult. For high-dimensional dynamical systems, such as 3D humanoid characters, mathematical analysis is intractable. Hence, simplified low-dimensional models that capture essential aspects of motion have a long history in biomechanics [Alexander 1980; Srinivasan and Ruina 2006] and locomotion control [Coros et al. 2010; da Silva et al. 2008a; Kajita et al. 2001; Kajita et al. 2003a; Pratt and Pratt 1998b; Pratt et al. 2001; Raibert 1986; Raibert and Hodgins 1991; Tsai et al. 2010; Yin et al. 2007].

Two basic models that have been proposed to explain the different patterns of time-varying forces measured during walking and running are the inverted pendulum model (IPM), and the spring-loaded inverted pendulum (SLIP) [Dickinson et al. 2000; Full and Koditschek 1999].

The IPM [Miura and Shimoyama 1984; Kajita and Tani 1991], models center-of-mass (COM) motion during walking. During walking the COM moves along arc-like trajectories about the center-of-pressure (COP). In the first half of the stance phase kinetic energy is transformed to gravitational potential energy. Kinetic energy is partially recovered in the second half of stance, as the COM moves forward and downward. The IPM assumes a constant leg-length, is constrained to have only one foot on the ground at all times, and does not model swing leg behavior. Hence, it cannot model double-stance, ballistic motion, or sequences of multiple motion phases.

The SLIP model [Full and Koditschek 1999; Schwind and Koditschek 2000] generalizes the IPM by replacing the fixed-length leg with a spring. This capture the body's ability to store energy, in muscles, tendons, and ligaments, during running. This energy storage and release process makes running analogous to bouncing on a pogo stick [Dickinson et al. 2000; Raibert 1986]. Despite their simplicity, exact SLIP equations-of-motion are non-integrable and are, therefore, costly to evaluate. In Chapter 5, we describe a closed-form SLIP-based model that captures motion across multiple phases, selects footholds producing good subsequent steps, and generates a continuum of behaviors including walking, running, and jumping.

The IPM and SLIP are, arguably, the simplest models for analysis of walking. In robotics and biomechanics, there is also considerable interest in understanding the role of passive dynamics in walking. The primary motivation of this work is to understand, and replicate, the energy efficiency of human walking.

In his seminal work, McGeer [1990; 1993] designed and analyzed a series of simple planar walking machines that walked down shallow slopes, despite having no active control or actuation. He showed that favorable mechanical design and minimal energy input, to compensate for impact losses, are sufficient to generate gaits approaching the efficiency and fluidity of humans. Since then, many researchers have sought to add actuation and control to these passive walkers, to make them suitable for flat ground locomotion (e.g., [Collins et al. 2005]).

Despite their relative simplicity, analyzing motion of passive walkers as a means of developing generalizable theories of locomotion remains challenging. These systems are highly nonlinear and rely on limit cycle dynamics for gait generation; results are very sensitive to system initial conditions. Extending dynamic walking to rough terrain is an active area of research [Byl and Tedrake 2008b; Byl and Tedrake 2008a].

To date, the IPM and SLIP remain the most tractable models of walking. In the following section, we detail locomotion control techniques based on these models. Despite many differences in the underlying details of how they are used they share a common goal: maintaining balance.

2.2 Maintaining Balance

> Life is like riding a bicycle. To keep your balance you must keep moving.
>
> Albert Einstein, 1879-1955

Bipedal locomotion requires balance. Two important factors influencing balance are: i) swing foot placement and ii) stance foot contact forces.

Selecting good swing foot targets is critical for smooth forward progress during locomotion and disturbance rejection. As anyone who has been pushed can attest to, good foot placement can be the difference between regaining balance, after a push, and speeding up, slowing down, or falling over. During locomotion this process can be exploited to regulate locomotion speed.

This straightforward insight is at the heart of the seminal work of Raibert, Hodgins, and colleagues [Raibert 1986; Raibert and Hodgins 1991; Hodgins et al. 1995; Hodgins and Pollard 1997]. For running and hopping gaits, they propose a three-part decomposition of locomotion control that regulates hopping height, hopping speed, and torso pitch, using decoupled control laws. They demonstrate stride-to-stride forward speed regulation, by adjusting swing foot touchdown position, for a SLIP-like robot. Since its introduction, this idea has been used in a number of animation and robotics applications [Kuo 1999; Laszlo et al. 1996; Talebinejad 2000]. Because these control laws assume steady-state behavior (i.e., controlled quantities depend on timing of previous strides), they can have difficulty handling transient behavior, such as sudden transitions from fast running to in-place hopping.

Yin et al [2007] extend this approach using continuous linear feedback based on simplified IPM analysis, in their SIMBICON system. At every time-step swing leg hip angle, in the world reference frame, is adjusted based on the COM position and velocity, relative to the stance foot. This strategy produces robust locomotion, by indirectly regulating foot touchdown position. Since its introduction, it has been used in several animation systems, for a wide variety of gaits [Coros et al. 2009; Wang et al. 2009; Wang et al. 2010; Yin et al. 2008]. This feedback law is computationally inexpensive but requires retuning control parameters for different characters and motions.

To improve the generalization capabilities of SIMBICON across characters of different dimensions, Coros et al. [2010] propose a linear feedback law based on capture-point analysis [Pratt and Tedrake 2005; Pratt et al. 2006; Rebula et al. 2007]. The n-step capture point model predicts the set of n foot steps needed to slow a walker to a standing configuration. For the special 1-step case, IPM energy analysis yields a closed-form analytical model. Manipulating the target foot plant location, relative to the 1-step capture point, yields a speed control mechanism analogous to the method of Raibert et al. [1986]. This method is limited to walking motions and does not directly apply to running or jumping.

In Chapter 5 we describe a method for optimal foot plant selection based on nonlinear optimization of a SLIP-based model. By considering future outcome of the simulation, we ensure that foot steps yield good results. Since the model is not based on gait-specific analysis, it applies to many locomotion gaits.

While swing foot placement is an important mechanism for controlling impulses applied at exchange of support, successful locomotion also requires regulating stance contact forces. Stance

forces are the byproduct of system inertial and dynamic properties, such as joint torques and accelerations, as well as mechanical properties of the environment, such as friction and elasticity. Hence, care must be taken to ensure that control torques do not lead to gait failure. For example, excessive ankle torques can cause the feet to roll about the toe or heel, leading to loss of balance.

To keep feet firmly planted during control, one alternative is to use model-based optimization [Abe et al. 2007; da Silva et al. 2008b; da Silva et al. 2008a; Hofmann et al. 2004; Jain et al. 2009; Macchietto et al. 2009; Muico et al. 2009]. By considering the coupled nature of dynamics and contact constraints, these approaches select optimal joint torques at every instant in time. Because these methods use an accurate system model, it is also possible to incorporate other types of constraints, such as limits on actuation and joints. The methods we describe in Chapters 3–5 fall into this category.

In addition to ensuring stance feet remain planted on the ground, regulation of stance contact forces can also be used for fine-scale velocity control. Zero Moment Point (ZMP) control [Vukobratovic and Juricic 1969] is a variant of this approach. By stiff tracking of carefully designed joint trajectories, the center-of-pressure can be kept within some "safe" stance foot region[1]. This approach yields cautious statically-stable gaits, differing significantly from dynamically-stable gaits employed by humans. Instead of trying to actively regulate the ZMP, an alternative is to use the ZMP to estimate gait stability. For example, monitoring the ZMP during locomotion can be used as an indicator the foot is rolling, or is about to roll [Popović et al. 2005]; preventative steps can be taken as needed [Wu and Zordan 2010].

2.3 Task-Space Control

To date, most existing locomotion controllers use joint-space representations. These methods use per-joint PD servos, coordinated by a high-level state machine [Faloutsos et al. 2001; Hodgins et al. 1995; Hodgins 1998; Laszlo et al. 1996; Raibert and Hodgins 1991; Wooten 1998; Yin et al. 2007]. This approach has been applied to a wide variety of walking, running, and athletic motions. The combination of high-gain tracking and discrete state machines frequently leads to motion that is difficult to tune for natural-looking full-body behaviors.

[1] For horizontal contact surfaces the ZMP and COP are equivalent [Popović et al. 2005]. While the former is computed from joint accelerations and mass properties, the latter is computed by direct measurement of contact forces.

As robots and simulated characters become more complex, there is increased interest in defining control in terms of high-level motion features. For example, for humanoid robotics, one might wish to strictly maintain balance while attempting to reach a target. Strong evidence is emerging to suggest that such low-dimensional control strategies are exploited by animals to manage complexity during control. Humans have been shown to only correct deviations in motion directly interfering with tasks [Todorov and Jordan 2002], while high-level similarities have been reported across a wide variety of animals during locomotion [Full and Koditschek 1999]. Task-level control decomposition can also make the application of learning methods (Section 2.6) tractable [Shkolnik and Tedrake 2008].

Task-space control has a long history [Abe and Popović 2006; Hsu et al. 1989; Khatib 1987; Liegeois 1977; Nakamura et al. 1987; Shkolnik and Tedrake 2008]; for a survey, see [Nakanishi et al. 2008]. Task-space strategies aim to achieve tasks as closely as possible, without lower-priority tasks interfering with higher-priority tasks, while also resolving ambiguities due to underdetermined tasks. Task-space strategies in the literature can be categorized by whether control is specified in terms of velocity [Liegeois 1977], acceleration [Hsu et al. 1989], or force [Khatib 1987]. These methods are based on null-space projection operators.

Velocity-based techniques gained early adoption because of their ease of implementation and modest computational requirements. These methods work well when high-gain tracking is desired, but are ill-suited for low-stiffness control [Nakanishi et al. 2008]. Acceleration and force level approaches are better suited for impedance control, but are more challenging to implement since they require a dynamic model of the system. Force-based methods, such as Operational Space Control (OSC) [Khatib 1987; Khatib et al. 2004] correctly decouple primary and secondary behaviors and ensure minimal interference between tasks. However these methods can require a potentially complex set of task-specific projection matrices and of their derivatives. Sentis [2007] extends OSC to whole-body motion for simulated humanoid robots, demonstrating a number of balancing and locomotion behaviors that handle numerous competing goals simultaneously.

An alternative force domain approach, first proposed by Sunada et al. [1994] and later formalized into Virtual Model Control (VMC) by Pratt et al. [1995; 2001], uses the Jacobian transpose (JT) mapping of forces to joint torques, derived from principles of virtual work [Craig 1989]. JT/VMC assumes static equilibrium; inertial properties, gravitational and Coriolis forces are neglected; redundancy and task conflict is not explicitly handled. Coros et al. [2010] use this

approach for gravity compensation, and COM velocity regulation. To allow task combination by superposition they are designed to not interfere with one another. Wu and Popović [2010] use optimization to resolve task conflicts.

Most null-space projection methods do not handle unilateral constraints; doing so is extremely complex and restricted to special cases [Mansard and Khatib 2008]. For restricted applications, iterative extensions to projection methods have been proposed (e.g., [Baerlocher and Boulic 2004]) that handle unilateral constraints by solving problems multiple times until all violations are eliminated. No performance guarantees exist for these methods.

To avoid calculating null-space projections, several methods formulate multiobjective optimizations that combine tasks into a single Quadratic Program (QP) [Abe et al. 2007; Azevedo et al. 2002; da Silva et al. 2008a; da Silva et al. 2008b; Fujimoto et al. 1998; Kudoh et al. 2006; Macchietto et al. 2009; Wu and Popović 2010] or non-convex optimization [Jain et al. 2009]. This avoids the need for projection operators, but cannot guarantee strict prioritization. These methods achieve impressive results in highly-constrained balancing and reaching scenarios. Because most QP methods use weighted combinations of objectives, this can results in objective terms that "fight" against each other. This makes it challenging to adjust weights in complex situations.

Instead of combining objectives in a weighted manner they can also be arranged and solved hierarchically, by order of importance. This scheme is referred to as lexicographic [Isermann 1982; Marler and Arora 2004; Rentmeester et al. 1996; Zykina 2003] or prioritized [Kanoun et al. 2009] optimization. Prioritized optimization (PO) ensures that objectives are achieved optimally, with minimal interference between objectives. To date, research on PO has focused primarily on nonlinear optimization problems; application of PO has received limited attention [Clark et al. 2008; Kanoun et al. 2009]. Most PO algorithms enforce higher-priority objectives through equality constraints (Section 6.3.4). In Chapter 6 we propose two PO algorithms, for quadratic programs, that use an alternate approach, based on solving each optimization in the null-space basis of all previous, higher-priority, problems. In our experience this scheme provides improved stability for task-space control applications.

2.4 Motion Capture Tracking

Designing control for highly stylized, physically-simulated, motion is very difficult. To ease this task, many methods have sought to leverage databases of recorded human motion capture (MOCAP) reference data. Although MOCAP data preserves many of the nuanced details that make motion feel "alive", using it for physics-based control poses several challenges.

For complex actions, such as walking, direct MOCAP playback through joint PD servos, results in rapid loss of balance (i.e., the character falls over after just a few steps). This is due to discrepancies between the recorded human subject and the physical model of the simulated character. Simulated models have fewer degrees of freedom, assume perfectly rigid links, and ideal revolute-joints; ground contact models make many approximations and often use ad-hoc hand-tuned parameters; precise *in vivo* measurement of human inertial properties are difficult to obtain.

To improve the robustness of MOCAP tracking approaches, several authors have explored using displacement maps. Displacement maps adapt MOCAP data by applying time-varying offsets to recorded trajectories. For planar actions, Sok and colleagues [2007] learn displacement maps using simplex optimization. Learned displacement maps are generalized using regression and composed into behavior-specific state machines. Parallel stochastic search can also be used to find displacement maps for complex 3D actions [Liu et al. 2010]. In both cases, learnt policies are open-loop; control does not have feedback. An alternative to displacement maps is to augment MOCAP tracking with some form of balance control; an arbitration mechanism is often required to avoid controller interference. A simple strategy to handle control conflicts is to assign different functional roles to different joints (e.g., lower-body handles balance, while upper-body handles tracking [Safonova et al. 2003; Zordan and Hodgins 2002]). Alternate approaches include hand-designed control [Yin et al. 2007] and simplified models [Tsai et al. 2010].

In recent years, many authors have turned to constrained optimization to resolve tracking and balance conflicts [Abe and Popović 2006], while also incorporating feedback. Feedback strategies based on finite-horizon planning [da Silva et al. 2008b], optimal control [da Silva et al. 2008a; Kwon and Hodgins 2010; Muico et al. 2009; Yamane and Hodgins 2009], and momentum compensation [Macchietto et al. 2009] have been proposed.

One of the main motivations for using physical simulation, for animation applications, is to generate results that adapt realistically to new environments and interactions. Although MOCAP makes it much easier to specify motion style, MOCAP-based control is typically robust only near reference trajectories (or transformations of them); new trajectories are required for every desired type and style of motion.

Many researchers are actively investigating techniques for adapting MOCAP-based control to new environments, body shapes, and unexpected perturbations. One approach is to lower PD gains following perturbations [Allen et al. 2007; Zordan and Hodgins 2002]; optimal control formulations based on Differential Dynamic Programing (DDP) has also been proposed [Ye and Liu 2010]. These approaches can produce qualitatively satisfying results, but have limited predictive capabilities. There is growing interest in developing biomechanically motivated adaptation strategies [Shiratori et al. 2009].

2.5 Interactive Control

Increases in computational power, have recently made interactive authoring of physics-based character motion possible. These techniques attempt to bypass the control design process, relying instead on the ability of humans to quickly learn and repeat motor tasks. An interesting application of interactive control is to provide training data for imitation-based learning of control policies [Loken 2006] (Section 2.6).

Using interactive input devices to control physically simulated characters presents a number of challenges. Unlike kinematic characters that do not model momentum, physically simulated characters cannot always respond to user commands instantaneously. For example, before executing a sharp turn, a running character must ensure that its foot is firmly planted on the ground; waiting for viable transition conditions can create a lag in system response.

Impressive acrobatic maneuvers and stunts can be authored for simplified planar characters that do not need to maintain balance and provide straightforward mappings from game controller inputs to character joints [Laszlo et al. 2000]. Interactive control of full 3D characters is more complex; numerous joints must be coordinated to avoid exceeding game controller inputs. Zhao and van de Panne [2005] demonstrate such a technique for challenging snowboarding motions. Shiratori and Hodgins [2008] investigate commanding transitions between preexisting

controllers, using Wiimote-based interaction techniques. When interactively controlling aggressive motion, with limited margin for error (e.g., flips), visualization of future system state can inform action selection [Laszlo et al. 2005].

2.6 Learning Methods

> Learning is not compulsory ... neither is survival.
>
> W. Edwards Deming, 1900-1993

Many theories of motor control are expressed in terms of optimality principles. This is appealing since many of the processes that give rise to human motion appear, in some sense, to be optimization processes; evolution, learning, and adaptation all help improve system performance over time [Alexander 1980; Alexander 1993; Bramble and Lieberman 2004; Harcourt-Smith and Aiello 2004; Todorov and Li 2003].

In the context of robotics and graphics, learning methods offer the long-term promise of principled automated control synthesis, with improved generalization capabilities, without need for complex dynamical modeling. In recent years, this has been an active area of research [Morimoto et al. 2004; Nakanishi et al. 2003; Sharon and van de Panne 2005; Smith 1998; Sok et al. 2007; Tedrake et al. 2004; Wang et al. 2009; Wang et al. 2010].

Learning approaches can be categorized into three main groups: i) supervised, ii) reinforcement, and iii) unsupervised learning.

Supervised Learning. In supervised learning, the system is presented with a series of inputs and the corresponding desired outputs. The goal of the learning system is to build a model mapping inputs to provided outputs. System performance is determined by measuring errors between actual and desired outputs. Two supervised learning approaches that have been used to date for locomotion control are imitation learning and feedback error learning.

Imitation learning [Atkeson and Schaal 1997; Loken 2006] uses recorded expert performances of behaviors to provide exemplar output. MOCAP tracking methods (Section 2.4) can be viewed as a variant of this approach. Although MOCAP data provides action-specific joint angles, corresponding internal forces are typically not observable. Since these are the quantities that must,

ultimately, be learned, this poses a significant problem.

An alternative is feedback error learning (FEL). For cyclic motion, FEL can be used to learn open-loop *feed-forward* control. It is conjectured that feed-forward control is a key strategy used by humans to generate low-stiffness motion [Kawato 1990; Nakanishi and Schaal 2004; Yin et al. 2007] and handle large delays in the central nervous system [Takahashi et al. 2001].

Reinforcement Learning and Optimal Control. In reinforcement learning (RL) [Sutton and Barto 1998], the system provides a measure of reward, at every instant in time, for every input and output. The goal of the learning system is to maximize the sum of total future rewards. An optional weighting can be used to trade-off between immediate and long-term gain. No exemplar output is provided in RL; the system simply specifies if the overall behavior is desirable. A major challenge of applying RL to non-trivial problems is the *temporal credit assignment* problem: for an outcome that depends on many past actions, which actions should be rewarded or penalized?

Most RL techniques proposed to date use little or no system knowledge; they attempt to find the globally optimal policy for a given task. Proposed methods include: Dynamic Programming [Bellman 1957; van de Panne et al. 1990], Genetic Algorithms [Allen and Faloutsos 2009; Auslander et al. 1995; Ngo and Marks 1993; Pratt and Pratt 1998a; Sims 1994; van de Panne and Fiume 1993], Generate-and-Test [van de Panne et al. 1994; Grzeszczuk and Terzopoulos 1995], Simulated Annealing [Wu and Popović 2003], Downhill Simplex [Tedrake and Seung 2002], Stochastic Gradient Descent [Tedrake et al. 2004; Sharon and van de Panne 2005], and Continuation methods [van de Panne and Lamouret 1995; Yin et al. 2008]. These approaches can work reasonably well on simple problems, but scale poorly. Methods generally assume discrete state and actions spaces, which scale exponentially as finer discretizations are used.

To improve performance, several methods leverage knowledge of system dynamics. These formulations have a long history in the engineering field of Optimal Control. Rooted in the calculus of variations, Optimal Control methods assume continuous state and action spaces.

One of the most famous results of Optimal Control theory is the Linear Quadratic Regulator (LQR). For linear time-invariant systems with quadratic cost, globally optimal linear feedback policies can be generated using the LQR formalism [Brotman and Netravali 1988; da Silva et

al. 2008a; Muico et al. 2009]. In practice, assumptions of linear dynamics only hold for trivial systems; more complex problems required dynamics to be linearized about a particular region of state-space. Hence, resulting policies are only locally optimal.

LQR-based schemes have become popular for optimal tracking of motion capture reference data (Section 2.4). Calculating LQR control for complex character at interactive rates remains computationally prohibitive. Hence, policies are typically precomputed about target reference trajectories [da Silva et al. 2008a; Muico et al. 2009]. Since resulting policies are based on linearized dynamics, they tend to generalize poorly far from reference motion. To address this problem, a number of policy composition methods have been proposed [da Silva et al. 2009; Todorov 2009; Tedrake 2009].

A closely related technique to Optimal Control is spacetime constraints. Spacetime constraints methods formulate motion synthesis as a constrained nonlinear optimization, whose goal is finding minimal energy motion trajectories [Fang and Pollard 2003; Witkin and Kass 1988]. Constraints enforce physical realism, contact events, and permit high-level artistic direction.

Spacetime constraints can produce impressive results for motions dominated by dynamics, such as running and acrobatic maneuvers. However, finding good solutions to low-energy motions with many local minima, such as walking, has proved more difficult. Several parameterization of spacetime constraints have been proposed to address this issue [Cohen 1992; Liu et al. 1994; Liu and Popović 2002; Liu et al. 2005; Safonova et al. 2004]. Spacetime constraints does not generate feedback control; motion is open-loop and can fail if the character is perturbed. Muico et al. [2009] augments spacetime with feedback mechanisms to improve robustness.

Spacetime constraints and LQR methods act optimally over some large time-horizon. An alternative is to act optimally, without considering future impact of actions (i.e., control is only optimal relative to the current system state). Problems of this type, with quadratic objectives and constraints linear in free variables, can be optimized using Quadratic Programming (QP) [Abe et al. 2007; da Silva et al. 2008b; da Silva et al. 2008a]. Although QP methods do not generate feedback policies, solvers are sufficiently fast that solutions can be recomputed at interactive rates.

Unsupervised Learning. In unsupervised learning, the system receives input from the environment but there is no measure of reward. For most varieties of unsupervised learning, the

desired output is the same as the input. This form of learning is most commonly used to build reduced dimensional models. To date, unsupervised learning has been used to perform Principal Component Analysis (PCA) [Wolpert et al. 2001], and identify low-dimensional motor primitives [Todorov and Ghahramani 2003].

Challenges. Although it is theoretically straightforward to use the above described learning methods, many practical difficulties must be overcome before they can be applied to non-trivial real-world problems. Selecting an appropriate controller representation is very challenging; representations that are too task-specific generalize poorly, while overly abstract representations may be difficult to train and can produce unexpected output. Abstract representations, such as function approximators, also scale poorly and are challenging to apply to high-dimensional state/action spaces, as encountered in motor control. Commonly used motor learning policy representations include: Neural Networks [Grzeszczuk et al. 1998; Sims 1994; van de Panne and Fiume 1993], Nearest-Neighbour [Coros et al. 2009; Sharon and van de Panne 2005], Locally-Weighted Projective Regression [Vijayakumar et al. 2005; Loken 2006], Pose Control Graphs [van de Panne et al. 1994], and manually-designed policies [Tedrake and Seung 2002; Yin et al. 2008; Wang et al. 2009; Wang et al. 2010; Wu and Popović 2003; Wu and Popović 2010].

Designing good objective functions is also difficult, particularly for animation applications, that seek to produce "natural" results, while also providing a high-degree of user control. Only a handful of techniques [Liu et al. 2005; Wang et al. 2009; Wang et al. 2010] propose objectives for generating "human-like" locomotion without resorting to motion-capture [Sok et al. 2007; Liu et al. 2010] or specialized mechanism design [Tedrake et al. 2004].

Chapter 3

Feature-Based Control

> Animation is not the art of drawings that move but the art of movements that are drawn
>
> Norman McLaren, 1914-1987

One method of generating physically-plausible character animation is to use dynamic simulation. Dynamic simulation models the character's dynamics, as well as the contact forces acting between the character and the environment. To make the character move in a desired fashion, dynamic simulation requires that internal forces (e.g., joint torques) be specified at runtime. This requires control. Specifying control for complex characters, with many joints, is difficult; many previous methods calculate torques based on the desired behavior of each individual joint.

In this chapter we describe feature-based control, an optimization-based approach for controlling physically simulated characters. The key idea of feature-based control is to parameterize desired behavior in terms of a small set of abstract quantities. In this thesis, we investigate features that are linear functions of free variables $\mathbf{x}$. Although free-variables vary between problems, for many examples:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}} \\ \mathbf{f}_c \end{bmatrix} \tag{3.1}$$

where $\boldsymbol{\tau}$ denotes joint torques, $\ddot{\mathbf{q}}$ denotes joint accelerations, and $\mathbf{f}_c$ are contact forces. We begin by summarizing key steps needed to design control using the feature-based representation; in the sections that follow we provide additional details.

The first step in designing a feature-based controller is to determine the task-relevant quantities to control. In practice, we find that many human motions can be described in terms of center-of-mass (COM), angular momentum (AM), and end-effector (EE) motion. For example, as we show in Chapter 4, walking can be specified in terms of just a few features: propel the COM forward and move the feet, while minimizing AM and torques for balance, stability, and style.

Once features have been selected, the next step is to specify desired reference, or target, values for each feature. Reference values may be fixed or may be time-varying trajectories. Each feature is then expressed, mathematically, as a quadratic objective that reflects how well a feature matches the desired reference value. In Section 3.2 we provide mathematical details for a number of objectives we use for motion design.

Next, each objective is assigned a priority expressing its importance with respect to all other objectives. Multiple objectives can also be combined at the same priority level. In our examples we typically place contact objectives at the highest level of priority. This reflects the fact that loss of balance is typically catastrophic for humans and that it is not possible to alter whole-body momentum without external forces.

Lastly, the set of objectives and priorities is given to a prioritized optimization (PO) algorithm (Chapter 6) that solves for $\mathbf{x}$ minimizing objectives and satisfying top-level equality and inequality constraints, which we denote as:

$$C(\mathbf{x}) = \mathbf{0} \tag{3.2}$$

$$\mathbf{D}\mathbf{x} + \mathbf{f} \geq \mathbf{0}. \tag{3.3}$$

In our system, we use equality constraints (3.2) to ensure optimization solutions are physically valid, while inequality constraints (3.3) restrict solutions such that: motion remains within valid ranges of joint movement, actuation limits are not violated, and contact forces do not exceed friction limits. Sections 3.3 and 4.4.2 provide details of terms contained in Equations 3.2 and 3.3.

We now proceed to give additional mathematical details of our simulation system, providing the needed background to develop our feature-based formulation. In this chapter we illustrate the flexibility of the feature-based formulation, using a set of simple examples. In subsequent chapters we apply feature-based control to more challenging tasks, including 3D humanoid balancing, standing jumps, walking, running, and uneven terrain navigation.

3.1 Dynamics and Simulation

In our system, a character is a series of interconnected rigid bodies, or links. Each link is assigned physical properties such as mass, moment-of-inertia, and a center-of-mass. Between each pair of rigid bodies, a joint is included. The number and orientation of the joint's degrees-of-freedom determine how links move relative to one another.

For bipedal humanoid characters, the described representation results in an open tree-like link topology, or "skeleton", with n_j joints and $n_l = n_j + 1$ links. A coordinate frame is attached to each link to provide a reference for link-local quantities (e.g., center-of-mass), thus avoiding many numerical issues. A fictitious "world" link is also included at the top of the hierarchy that acts as the root of the joint/link hierarchy and provides a fixed reference frame.

For each joint i, we can define a vector of generalized coordinates positions $\mathbf{q}^i$ and its time-derivative (i.e., velocity) $\dot{\mathbf{q}}^i$. Assuming inertial properties remain constant, and that links are completely rigid, it is possible to fully describe the character's instantaneous state by aggregating all joint information into the high-dimensional vectors:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}^0 \\ \vdots \\ \mathbf{q}^{n_j} \end{bmatrix} \qquad \dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}^0 \\ \vdots \\ \dot{\mathbf{q}}^{n_j} \end{bmatrix} \tag{3.4}$$

where $\mathbf{q} \in \mathbb{R}^{n_p \times 1}$, $\dot{\mathbf{q}} \in \mathbb{R}^{n_v \times 1}$, and n_p/n_v are the total number of position/velocity degrees-of-freedom respectively. For the systems described in this thesis $n_p \geq n_v$.

Since we are interested in generating animation, we need to calculate how the character's state changes over time; we do this using a model based on Newtonian dynamics. For a given state $(\mathbf{q}_t, \dot{\mathbf{q}}_t)$, at time t, and applied forces $\mathbf{u}_t \in \mathbb{R}^{n_v \times 1}$, we derive equations-of-motion:

$$\ddot{\mathbf{q}}_t = f(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{u}_t) \tag{3.5}$$

that calculate generalized coordinate accelerations. Integrating these accelerations provides an estimate of the system state in the future. For example, using Euler integration:

$$\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}_{t+\Delta t} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}_t + \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix}_t \Delta t. \tag{3.6}$$

Many more complex integration schemes exist, each offering different benefits; see [Press et al. 2007] for a more complete discussion.

Generating equations-of-motion (Equation 3.5) efficiently for articulated rigid-body dynamical systems is a problem that has a long history in robotics and applied mechanics; many techniques exist to solve this problem.

One possibility is to express the equations-of-motions, in manipulator form:

$$\mathbf{u}_t = \mathbf{M}(\mathbf{q}_t)\ddot{\mathbf{q}}_t + \mathbf{h}(\mathbf{q}_t, \dot{\mathbf{q}}_t)s \tag{3.7}$$

where $\mathbf{M}$ and $\mathbf{h}$ are the joint-space inertia matrix and Coriolis/gravitational forces respectively. $\mathbf{M}$ and $\mathbf{h}$ can be efficiently calculated using the Composite Rigid-Body and Recursive Newton-Euler algorithms [Featherstone 2008]. For brevity, we omit dependence of these quantities on $\mathbf{q}$ and $\dot{\mathbf{q}}$ in the remainder of the chapter. Equation 3.7 is simply a generalization of Newton's second law:

$$\mathbf{f} = m\,\mathbf{a} \tag{3.8}$$

from introductory physics, for more complex systems. Note that in order to solve for $\ddot{\mathbf{q}}$ using this approach we must invert $\mathbf{M}$. To obtain improved performance, methods exist that calculate $\ddot{\mathbf{q}}$, without explicitly computing or inverting $\mathbf{M}$, while also taking advantage of branch-induced sparsity, present in most articulated body systems. Our implementation uses the Articulated Rigid-Body Method [Featherstone 2008].

To calculate generalized accelerations, $\ddot{\mathbf{q}}$, using Equation 3.5, we need to specify forces, $\mathbf{u}$, acting on the system. Forces acting on a mechanical system:

$$\mathbf{u} = (\mathbf{f}_{int}, \mathbf{f}_{ext}) \tag{3.9}$$

can be classified as external or internal. External forces include gravity, contact forces, friction, and so on. In humans, internal forces come from muscles, tendons, ligaments, and cartilage. In our simulation, we approximate these biological force generation mechanisms; internal forces

come from torques:

$$\mathbf{f}_{int} = \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}^0 \\ \vdots \\ \boldsymbol{\tau}^{n_j} \end{bmatrix} \tag{3.10}$$

applied at each joint, where $\boldsymbol{\tau} \in \mathbb{R}^{n_v \times 1}$.

For floating-base systems, such as humanoids, the joint connecting the character to the world, $\mathbf{q}^0 \in \mathbb{R}^{6\times 1}$, is passive (i.e., $\boldsymbol{\tau}^0 = \mathbf{0}$). This captures the fact that humans are underactuated and can only generate motion through internal forces. At each instant in time, remaining joint torques:

$$\hat{\boldsymbol{\tau}} = \begin{bmatrix} \boldsymbol{\tau}^1 \\ \vdots \\ \boldsymbol{\tau}^{n_j} \end{bmatrix} \tag{3.11}$$

are generated using a control algorithm or *controller*. More generally, a controller is a mapping:

$$\hat{\boldsymbol{\tau}} = f(\mathbf{q}, \dot{\mathbf{q}}, t) \tag{3.12}$$

from character state to torques. Controller torques can also depend on other quantities, such as time, internal controller state, contact forces, and mocap reference motion recordings.

A common approach is to parameterize control in terms of individual joint actions, as:

$$\boldsymbol{\tau}^i = k_p(\mathbf{q}_r^i - \mathbf{q}^i) + k_v(\dot{\mathbf{q}}_r^i - \dot{\mathbf{q}}^i) \quad i > 0 \tag{3.13}$$

using proportional-derivative (PD) control[1]. Because individual joint actions combine in a highly non-linear and interdependent way, expressing coordinated or stylized motion in this manner is extremely difficult. For example, tuning PD gains to obtain particular motion styles can require iteration; results depend not only on parameters in Equation 3.13 but also on the contact state of different limbs. Goals of generating motion in a particular style may also conflict with requirements for accurate tracking of reference quantities; increasing k_p reduces po-

[1]This equation assumes all joints are one degree-of-freedom revolute joints. Special care must be taken to correctly calculate positional error for spherical/ball joints represented using quaternions [da Silva et al. 2008b]

sitional error at the expense of yielding "stiff" results, while increasing k_v can make motion appear "soupy". For in-place gesturing motions an alternative is to specify gains by reasoning about forces in terms of antagonistic forces [Neff and Fiume 2002].

Instead of specifying control torques in terms of individual joint actions, feature-based control solves for torques by formulating an optimization in terms of N quadratic objective functions, $E_i(\mathbf{x})$, in unknowns $\mathbf{x}$ (Section 3.3) subject to dynamics and control specific constraints. Each objective expresses some characteristic of the desired motion.

To solve this optimization, we use the prioritized optimization algorithms described in Chapter 6. This can be summarized as:

$$\begin{aligned} h_i = & \min_{\mathbf{x}} E_i(\mathbf{x}) \\ \text{subject to } & E_k(\mathbf{x}) = h_k,\ \forall k < i \\ & C(\mathbf{x}) = \mathbf{0} \\ & \mathbf{D}\mathbf{x} + \mathbf{f} \geq \mathbf{0}. \end{aligned} \tag{3.14}$$

Variables may vary with time t, which we omit from these equations for brevity. In other words, the first optimization problem is $h_1 = \min_{\mathbf{x}} E_1(\mathbf{x})$, subject to constraints. Subsequent steps are required to achieve this minimum: the second problem is $h_2 = \min_{\mathbf{x}} E_2(\mathbf{x})$, subject to $h_1 = E_1(\mathbf{x})$ and constraints. This optimization is repeated recursively, returning the point $\mathbf{x}_N$ that solves the N-th problem.

A key aspect of this approach is that because each feature we control is formulated as an objective, they will either be exactly satisfied or we will be given the minimum norm solution to our control problem. The latter case can only occur if a higher-priority objective interferes with a lower-priority objective. Hence, residuals can be calculated to determine to what degree the solution returned by the solver satisfies the specified control objectives.

3.2 Features and Objectives

In this section we summarize objectives used to design feature-based control. Each objective is formulated in terms of some feature $\mathbf{y}^i$ of the motion, such as COM, foot position, AM, joint torque, or pose. We use 4 different types of objectives to control these features: i) Setpoint, ii)

Target, iii) Angular Momentum, and iv) Minimum Torque objectives. We now describe each of these objectives.

3.2.1 Setpoint Objective

This objective applies linear control to some feature of pose. In this case, a feature is any algebraic function:

$$\mathbf{y}^i = f(\mathbf{q}) \tag{3.15}$$

of generalized coordinates $\mathbf{q}$. To define a setpoint objective, we first specify "reference" values $\mathbf{y}^i_r, \dot{\mathbf{y}}^i_r, \ddot{\mathbf{y}}^i_r$ for the feature. A desired acceleration is computed by linear control:

$$\ddot{\mathbf{y}}^i_d = k_p(\mathbf{y}^i_r - \mathbf{y}^i) + k_v(\dot{\mathbf{y}}^i_r - \dot{\mathbf{y}}^i) + \ddot{\mathbf{y}}^i_r \tag{3.16}$$

where k_p and k_v are feature-specific gains. Unless noted otherwise, we set $k_v = 2\sqrt{k_p}$ in our examples. For an ideal second-order spring-damper system of unit mass, this yields a critically-damped response to step changes in reference values, $\mathbf{y}_r$ [Thomson 1993].

The setpoint objective then measures the deviation of the feature acceleration from the desired acceleration:

$$E_i(\mathbf{x}) = \left|\left| \ddot{\mathbf{y}}^i - \ddot{\mathbf{y}}^i_d \right|\right|^2 \tag{3.17}$$

where the actual feature acceleration $\ddot{\mathbf{y}}^i$ can be computed by differentiating Equation 3.15 twice:

$$\dot{\mathbf{y}}^i = \mathbf{J}_i\dot{\mathbf{q}} \tag{3.18}$$

$$\ddot{\mathbf{y}}^i = \mathbf{J}_i\ddot{\mathbf{q}} + \dot{\mathbf{J}}_i\dot{\mathbf{q}} \tag{3.19}$$

and

$$\mathbf{J}_i = \frac{\partial \mathbf{y}^i}{\partial \mathbf{q}} \tag{3.20}$$

is the Jacobian matrix of $\mathbf{y}^i$, and $\dot{\mathbf{J}}_i$ is its time derivative. For example, we may define a COM feature $\mathbf{y}^{com}$ and then servo it to a setpoint $\mathbf{y}^{com}_r$. This is normally used to maintain a feature

at a fixed value, (e.g., maintaining COM over the feet). Appendix B describes how to compute these Jacobians efficiently.

3.2.2 Target Objective

When we wish to servo a feature to a distant setpoint (i.e., $\mathbf{y}_r^i \gg \mathbf{y}^i$), directly applying the setpoint objective would cause large impulsive forces since Equation 3.16 is dominated by the positional error term. One alternative is to servo along a smooth time-varying trajectory $\mathbf{y}_r(t)$, (e.g., using Equation 3.16), with appropriate boundary conditions (e.g., $\mathbf{y}_r^i(0) = \mathbf{y}^i(0)$). However, this requires time-consuming trajectory design and servo-gain tuning.

To avoid this manual design step, we propose a method that directly computes $\ddot{\mathbf{y}}_d$ by solving a simple boundary value problem to move the feature from the current state $(\mathbf{y}_0, \dot{\mathbf{y}}_0)$ to a target state $(\mathbf{y}_T, \dot{\mathbf{y}}_T)$, in duration T. In practice, we use this type of objective to move a feature between two target locations, such as when the COM height is controlled during jumping (Section 4.2) or feet are guided between footholds in walking (Section 4.3).

Solving for each element of $\mathbf{y} \in \mathbb{R}^D$ separately, and assuming a linear solution of the form:

$$\ddot{y}_d(t) = \left(1 - \frac{t}{T}\right) a + \frac{t}{T} b \tag{3.21}$$

we seek constants a and b generating the desired motion. Integrating Equation 3.21 twice, isolating a and b, and writing in matrix form yields:

$$\begin{bmatrix} T^2/3 & T^2/6 \\ T/2 & T/2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_T - y_0 - \dot{y}_0 T \\ \dot{y}_T - \dot{y}_0 \end{bmatrix}. \tag{3.22}$$

At each instant of the simulation, we recompute a and b, by solving the above linear system in a least-square sense using a singularity-robust pseudoinverse (Appendix A.2). This avoids instabilities that may occur as $T \to 0$. For large values $T \gg 0$, this approach is equivalent to cubic Hermite interpolation; consult Appendix A.3 for a formal proof.

Then, the objective for each feature is:

$$E_{target}(\mathbf{x}) = \left(\ddot{y}_d(0) - \ddot{y} \right)^2. \tag{3.23}$$

This calculation determines the target acceleration independent of the object's mass. See Appendix A for a derivation of (3.22).

3.2.3 Angular Momentum Objective

Biomechanical observations of human motion reveal that AM about the COM[2] is very small during walking [Herr and Popović 2008; Popović et al. 2004]. Directly regulating AM during standing tasks, has also been shown to improve balance stability and robustness, via strategies such as windmilling [Macchietto et al. 2009; Kudoh et al. 2006].

Current AM regulation strategies (e.g., [Macchietto et al. 2009]) have several limitations. First, they require a reference center-of-pressure (COP) to be specified. Although a fixed COP is sufficient for static motions such as balancing, more complex actions (e.g., walking) require time-varying COP trajectories [Adamczyk et al. 2006]. We propose an alternate AM control formulation that produces good results not only for balance, but also jumping and walking. Second, methods rely on knowledge of contact forces prior to computing control, limiting the technique to simulators with penalty-based ground contact models. To avoid interpenetration and tuning of contact model parameters, modern simulators formulate calculation of contact forces as Linear Complementarity Problems (LCP) (Section 4.5). These simulators compute contact forces *after* joint-torques are specified. We were unable to make the method of [Macchietto et al. 2009] work in our LCP-based simulator. Our method does not rely on knowledge of contact forces for AM regulation. Lastly, since we do not explicitly introduce coupling between linear and AM regulation, our approach is simpler to implement.

We propose a quadratic objective in free variables:

$$E_{AM}(\mathbf{x}) = \left|\left| \dot{\mathbf{L}} - \dot{\mathbf{L}}_d \right|\right|^2 \tag{3.24}$$

that measures changes in whole-body AM and specifies desired change in AM:

$$\dot{\mathbf{L}}_d = k_p(\mathbf{L}_r - \mathbf{L}) \tag{3.25}$$

using linear control. In Equations 3.24 and 3.25, $\dot{\mathbf{L}}$ is the rate of change AM about the COM and

[2] AM about the COM is also referred to as the spin angular momentum [Herr and Popović 2008; Popović et al. 2004] or centroidal angular momentum [Orin and Goswami 2008]

$\mathbf{L}_r$ is the reference AM about the COM. For balancing (Section 4.1) and walking (Section 4.3), we set $\mathbf{L}_r = 0$, which corresponds to damping rotations [Popović et al. 2004]. For acrobatic jumping maneuvers (Section 4.2), we specify non-zero $\mathbf{L}_r$ to create twisting behavior.

To calculate $\dot{\mathbf{L}}$, we express the AM about the COM as:

$$\mathbf{L} = \mathbf{P}\mathbf{J}_{AM}\dot{\mathbf{q}} \tag{3.26}$$

$$\mathbf{J}_{AM} = \begin{bmatrix} \mathbf{J}_1^T & \dots & \mathbf{J}_n^T \end{bmatrix}^T \tag{3.27}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_1 & m_1(\mathbf{p}_1 - \mathbf{c})\times & \dots & \mathbf{I}_n & m_n(\mathbf{p}_n - \mathbf{c})\times \end{bmatrix} \tag{3.28}$$

where $\mathbf{I}_i$ is the inertia of link i about the COM, m_i is the mass of link i, $\mathbf{p}_i$ is the position of link i's COM, $\mathbf{c}$ is the position of the system COM, $\mathbf{J}_i = \frac{\partial \mathbf{c}_i}{\partial \mathbf{q}}$ maps joint velocities to spatial velocities about each link's COM, and $\dot{\mathbf{q}}$ is the vector of joint velocities. Differentiating (3.26), we obtain:

$$\dot{\mathbf{L}} = \mathbf{P}\mathbf{J}_{AM}\ddot{\mathbf{q}} + (\dot{\mathbf{P}}\mathbf{J}_{AM} + \mathbf{P}\dot{\mathbf{J}}_{AM})\dot{\mathbf{q}} \tag{3.29}$$

$$\dot{\mathbf{P}} = \begin{bmatrix} \omega_1\times\mathbf{I}_1 - \mathbf{I}_1\omega_1\times & m_1(\mathbf{v}_1 - \dot{\mathbf{c}})\times & \dots & \omega_n\times\mathbf{I}_n - \mathbf{I}_n\omega_n\times & m_n(\mathbf{v}_n - \dot{\mathbf{c}})\times \end{bmatrix} \tag{3.30}$$

where ω_i is the angular velocity of the i^{th} link, $\mathbf{v}_i$ is the linear velocity of link i's COM, and $\dot{\mathbf{c}}$ is the velocity of the system COM. All of the above quantities are expressed in the world reference frame. Note that the product $\mathbf{P}\mathbf{J}_{AM}$ is equivalent to the first 3 rows of the Centroidal Momentum Matrix (i.e., $\mathbf{A}_G$) [Orin and Goswami 2008], and the bottom 3 rows of the large RHS matrix in Equation 1 of [Kajita et al. 2003b].

3.2.4 Minimum Torque Objective

We use an additional objective to minimize joint torques:

$$E_\tau(\mathbf{x}) = ||\,\boldsymbol{\tau}\,||^2 = \left|\left|\begin{bmatrix} \mathbf{S} & \mathbf{0} \end{bmatrix}\mathbf{x}\right|\right|^2 \tag{3.31}$$

where the selection matrix $\mathbf{S}$ is a square matrix of zeros, with ones on the diagonal for joints we want to make passive. We do this by penalizing application of control torques. It is used in Section 3.3.2 to make user-selected joints passive and to produce natural-looking arm sway in walking, by minimizing shoulder and elbow torques (Section 4.3).

3.3 Example Controllers

We now apply feature-based control to three control problems: tracking a target with a planar serial chain, tracking a target with an underactuated planar chain, and performing squats with a planar biped with bilateral ground forces. All examples are shown in the accompanying video.

The simulations range in complexity from 4 to 9 degrees-of-freedom, with the individual tasks ranging from 8 to 20 dimensions. All examples are integrated and controlled at 1 kHz and run in real time; dynamics are integrated as in Equation 3.6. Unconstrained prioritized optimization (Section 6.1) is used for all examples.

When defining a controller, one must define a set of objectives $E_i(\mathbf{x})$ that, at each time-step, are provided to the optimizer to calculate joint torques. We define the vector of free variables as:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}} \end{bmatrix}. \tag{3.32}$$

Substituting $\mathbf{u}_t = \boldsymbol{\tau}$ into Equation 3.7, and rewriting in terms of $\mathbf{x}$, yields dynamics equality constraints:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \end{bmatrix} \mathbf{x} - \mathbf{h} = \mathbf{0} \tag{3.33}$$

relating joint torques and accelerations.

We then define additional objectives for the features we wish to control. Note that the highest priority objective $E_1(\mathbf{x})$ will always be satisfied as long as it is achievable without violating equality constraints. Lower-level objectives, $E_i(\mathbf{x})$, will be satisfied as closely as possible as well. It is important to define sufficient features to completely specify torques. Although unconstrained prioritized optimization returns the minimum norm solution when ambiguities remain (e.g., Section 6.1, Algorithm 1, line 11), in practice this solution leads to instability due to insufficient damping. Instead, in all the examples that follow, we define the final objective such that it affects all torques and includes some dissipative component. For example, the final objective may damp velocities or servo all joints to some default pose. Since the feature will be full rank, it will not be exactly satisfied. Instead, we will obtain the solution satisfying the damping objective as closely as possible. Based on our experience, this is sufficient to prevent instability.

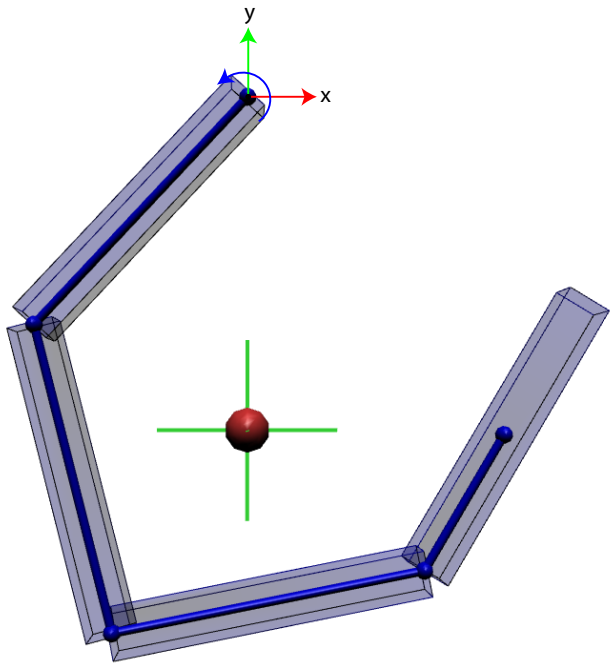

Figure 3.1: Planar Four Link Chain Simulation. The chain's actual COM location is shown as a red sphere, while the commanded (i.e., reference) COM location is shown as a green crosshair. Chain movement is restricted to the XY plane.

3.3.1 Fully Actuated Chain

A fully actuated planar serial-chain with four one degree-of-freedom revolute joints is first simulated (Figure 3.1). All links are given identical inertial and geometric properties (length=0.7 m, mass=10 kg, inertia=0.41 kg·m^2). The primary control objective is to servo the chain's COM:

$$\mathbf{y}^{com} = f(\mathbf{q}) \tag{3.34}$$

along a two-dimensional figure-eight reference trajectory constrained to the XY plane:

$$\mathbf{y}_r = \begin{bmatrix} A\sin(t) \\ A\sin(2t) \end{bmatrix}. \tag{3.35}$$

This is accomplished using a setpoint objective:

$$E_1(\mathbf{x}) = || \ddot{\mathbf{y}}^{com} - \ddot{\mathbf{y}}_d^{com} ||^2 \tag{3.36}$$

$$= \left|\left| \mathbf{J}_{com}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{com}\dot{\mathbf{q}} - \ddot{\mathbf{y}}_d^{com} \right|\right|^2 \tag{3.37}$$

where the desired acceleration $\ddot{\mathbf{y}}_d^{com}$ is determined according to Equation 3.16 and

$$\ddot{\mathbf{y}}^{com} = \mathbf{J}_{com}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{com}\dot{\mathbf{q}} \tag{3.38}$$

is obtained by differentiating Equation 3.34 twice with respect to time.

To resolve redundancy in the remaining degrees-of-freedom, a secondary objectives servos all joints to a reference posture:

$$E_2(\mathbf{x}) = \left|\left|\, \ddot{\mathbf{y}}^{posture} - \ddot{\mathbf{y}}_d^{posture} \,\right|\right|^2 \tag{3.39}$$

$$= \left|\left|\, \mathbf{J}_{posture}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{posture}\dot{\mathbf{q}} - \ddot{\mathbf{y}}_d^{posture} \,\right|\right|^2. \tag{3.40}$$

Since the second setpoint objective is for a configuration-space task:

$$\mathbf{y}^{posture} = \mathbf{q} \tag{3.41}$$

$$\dot{\mathbf{y}}^{posture} = \dot{\mathbf{q}} \tag{3.42}$$

$$\ddot{\mathbf{y}}^{posture} = \ddot{\mathbf{q}} \tag{3.43}$$

or equivalently,

$$\mathbf{J}_{posture} = \mathbf{I} \tag{3.44}$$

$$\dot{\mathbf{J}}_{posture} = \mathbf{0}. \tag{3.45}$$

Hence (3.40) can be written as:

$$E_2(\mathbf{x}) = \left|\left|\, \ddot{\mathbf{q}} - \ddot{\mathbf{y}}_d^{posture} \,\right|\right|^2. \tag{3.46}$$

Summarizing all constraints and objectives we obtain:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \end{bmatrix} \mathbf{x} - \mathbf{h} = \mathbf{0} \tag{3.47}$$

$$E_1(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{J}_{com} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \ddot{\mathbf{y}}_d^{com} - \dot{\mathbf{J}}_{com}\dot{\mathbf{q}} \end{bmatrix} \right|\right|^2 \tag{3.48}$$

$$E_2(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} - \ddot{\mathbf{y}}_d^{posture} \right|\right|^2. \tag{3.49}$$

Figure 3.2 shows commanded and actual COM coordinates in the XY plane for a 15 second sim-

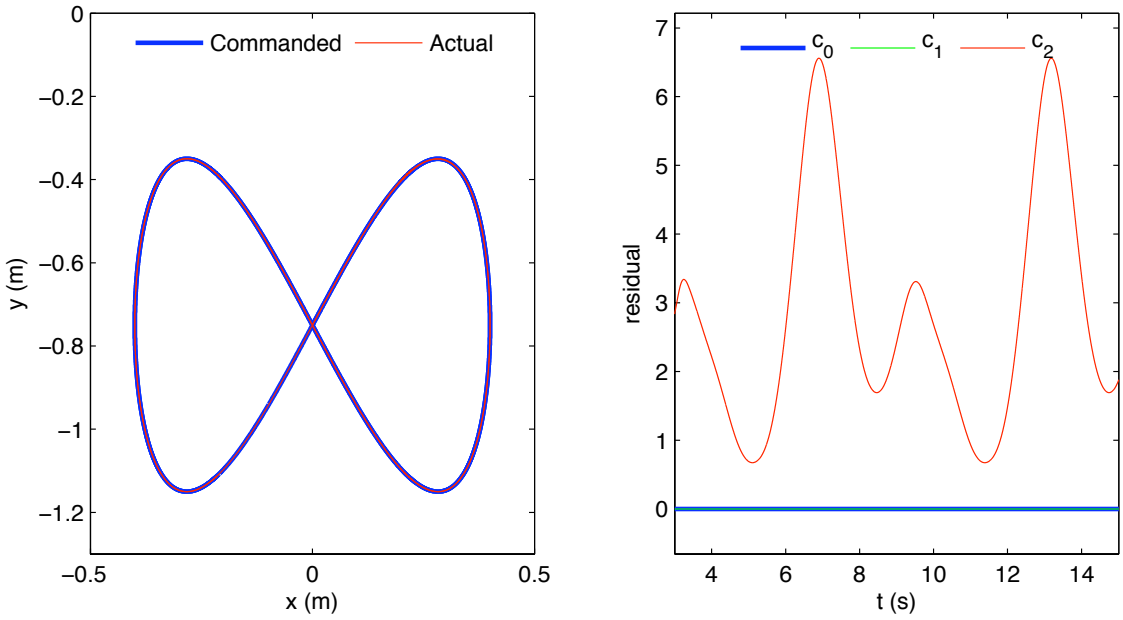

Figure 3.2: Planar Four Link Chain Results. Objective gains are E_1: $k_p = 100$, $k_v = 10$, E_2: $k_p = 0$, $k_v = 1000$. **Left:** COM commanded and actual trajectories, **Right:** Residuals. c_0: dynamics constraints, $i > 0, c_i = E_i$

ulation using the described controller; unconstrained prioritized optimization objective values are also plotted. As expected, the COM objective, E_1, is exactly satisfied (i.e., $c_1 = 0$), while the pose task objective, E_2, cannot be completely satisfied (i.e., $c_2 \neq 0$). This occurs since E_2 operates in the mechanism's configuration space and conflicts with the COM objective. c_0 corresponds to problem equality constraints, $C(\mathbf{x})$, which we enforce via a top-level objective E_0.

3.3.2 Underactuated Chain

A mechanism is said to be underactuated if internal forces cannot produce accelerations in certain degrees-of-freedom. This includes well-known systems such as Acrobot [Spong 1994] and all untethered robots and simulated characters. Underactuation arises when a system has more degrees-of-freedom than actuators.

One strategy used to control underactuated systems is to use a partial feedback linearization (PFL), to couple active and passive degrees-of-freedom [Shkolnik and Tedrake 2008; Spong 1994]. Due to the generality of our technique, defining a PFL for an underactuated version of the system requires only minor modifications to the specification. This avoids having to manually decompose the mass matrix into active and passive parts [Shkolnik and Tedrake 2008].

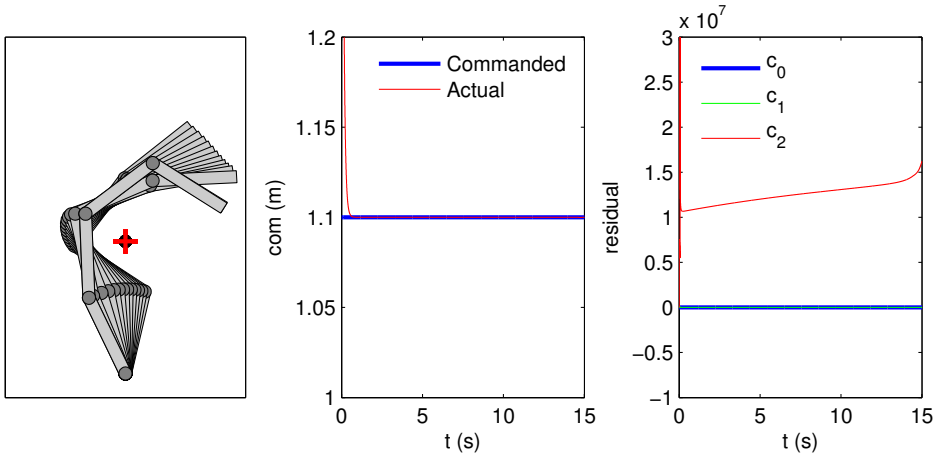

Figure 3.3: Underactuated Chain Example. Left: Time-series of underactuated chain during COM tracking task. Desired COM height shown as red crosshair. Actual COM height shown as black circle. **Middle:** Commanded (blue) and actual (red) values for COM height. **Right:** Prioritized optimization residuals for provided objectives.

In this example, we perform a setpoint regulation task with an underactuated version of the 4-link serial chain, from Section 3.3.1. To do this, we constrain the first joint to be passive (i.e., $\tau^0 = 0$), by augmenting the equality constraints with a minimum torque objective (Section 3.2.4). Hence, constraints and objectives are:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \\ \mathbf{S} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{h} \\ \mathbf{0} \end{bmatrix} = \mathbf{0} \tag{3.50}$$

$$E_1(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{J}_{com} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{y}_d^{com} - \dot{\mathbf{J}}_{com} \end{bmatrix} \right|\right|^2 \tag{3.51}$$

$$E_2(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} - \ddot{\mathbf{y}}_d^{posture} \right|\right|^2 \tag{3.52}$$

where

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.53}$$

is the joint selection matrix. We set $k_p = 0$ in E_2, such that the posture objective only applies damping forces. All other quantities/gains are unchanged from Section 3.3.1.

Figure 3.3 shows results for this simulation. Despite making the base joint passive, the COM height is accurately tracked. As can be seen from the solver residuals, with the exception of the final full-rank task, all task objectives are met.

3.3.3 Bilateral Constraints

To test how our method scales to more complex problems, we simulate a planar human performing squats (Figure 3.5). We use a HAT model (Figure 3.4) that aggregates head, arms, and trunk mass properties into a single link. Character height and weight correspond to a 50th percentile North American male, with individual link mass/geometric properties estimated using data from Winter [2004]. Given link mass and geometric properties, inertia is calculated using a thin-rod assumption (Table 3.1).

We model foot/ground interaction using a set of bilateral forces, acting at 2 contact points ($\mathbf{p}_{1r}$, $\mathbf{p}_{2r}$, $\mathbf{p}_{1l}$, $\mathbf{p}_{2l}$) on each foot (Figure 3.4). This requires that we include external forces, $\mathbf{f}_c$ as unknowns:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}} \\ \mathbf{f}_c \end{bmatrix} \tag{3.54}$$

and that we augment Equation 3.7 to account for the torques generated by these external forces:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{h} = \boldsymbol{\tau} + \mathbf{J}_c^T \mathbf{f}_c \tag{3.55}$$

where $\mathbf{J}_c^T = \begin{bmatrix} \mathbf{J}_{1r}^T & \mathbf{J}_{2r}^T & \mathbf{J}_{1l}^T & \mathbf{J}_{2l}^T \end{bmatrix}$ contains the Jacobians $\mathbf{J}_\star = \frac{\partial \mathbf{p}_\star}{\partial \mathbf{q}}$ for all contact points.

Three other setpoint objectives are provided in our example: an objective to regulate horizontal and vertical COM position, an objective to regulate trunk orientation, and a final configuration-space objective servoing all joints to a reference posture.

The complete set of constraints and objectives is:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} & \mathbf{J}_c^T \\ \mathbf{0} & \mathbf{J}_c & \mathbf{0} \\ \mathbf{S}_{pelvis} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{h} \\ \ddot{\mathbf{y}}_d^c - \dot{\mathbf{J}}_c \dot{\mathbf{q}} \\ \mathbf{0} \end{bmatrix} \tag{3.56}$$

$$E_1(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{J}_{com} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \ddot{\mathbf{y}}_d^{com} - \dot{\mathbf{J}}^{com} \dot{\mathbf{q}} \end{bmatrix} \right|\right|^2 \tag{3.57}$$

$$E_2(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{J}_{trunk} & \mathbf{0} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \ddot{\mathbf{y}}_d^{trunk} - \dot{\mathbf{J}}^{trunk} \dot{\mathbf{q}} \end{bmatrix} \right|\right|^2 \tag{3.58}$$

$$E_3(\mathbf{x}) = \left|\left| \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x} - \ddot{\mathbf{y}}_d^{posture} \right|\right|^2. \tag{3.59}$$

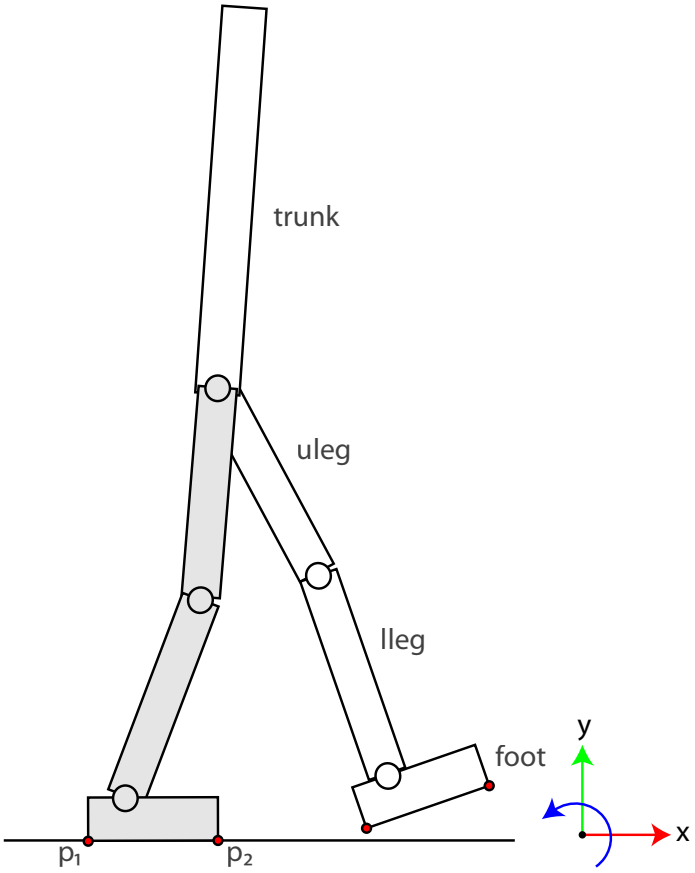

Figure 3.4: Planar HAT model for squatting simulations. Model has 7 links, 9 degrees-of-freedom with a floating base, connecting the character's pelvis to the world via a three degree-of-freedom planar joint. Inertial parameters are taken for a 50th percentile North American male.

In the above $\mathbf{S}_{pelvis} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times6} \end{bmatrix}$ is the passive joint selection matrix for our simulation. This objective ensures that no artificial forces are applied at the pelvis. We compensate for numerical drift arising from expressing contact point position constraints at the acceleration level using a setpoint objective on the contact points (3.56); this is equivalent to Baumgarte stabilization [Cline and Pai 2003].

Figure 3.6 summarizes key results for this example. As can be seen, COM position and trunk orientation are accurately tracked. With the exception of the last full-rank task, residuals for all other tasks are within machine tolerance of zero.

Table 3.1: Planar Biped Mass Properties. Provided COM values are expressed in link local coordinates. Inertia is expressed about each link's COM.

Link	Length $[m]$	Mass $[kg]$	COM (x,y) $[m]$		Inertia $[kg \cdot m^2]$
trunk	0.846	55.47	0.00	0.325	3.670
uleg	0.441	8.18	0.00	-0.191	0.166
lleg	0.443	3.81	0.00	-0.191	0.068
foot	0.274	1.19	0.07	-0.035	0.001
total	1.80	81.82			

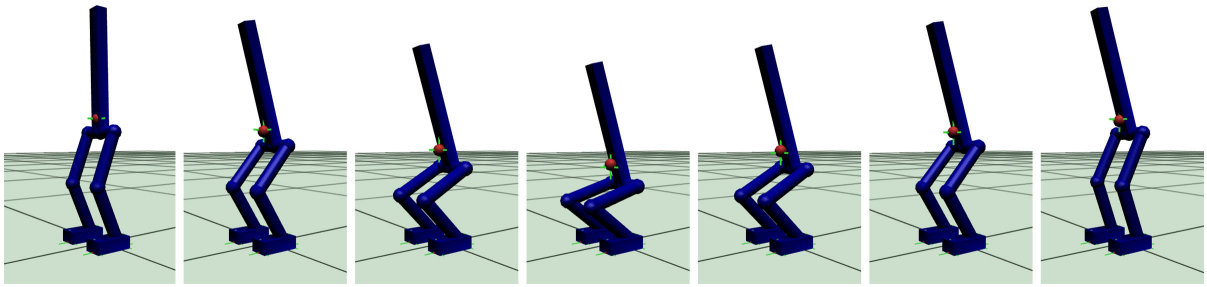

Figure 3.5: Seven frames from the planar biped squatting simulation. Actual and desired COM/contact point positions are shown as red spheres and green crosshairs respectively.

3.4 Comparison to Quadratic Programming

An alternative to executing tasks in strict priority order is to combine them in a weighted fashion:

$$\begin{aligned} \mathbf{x}^* = \arg\min_{\mathbf{x}} & \sum_{i=1}^{N} \alpha_i E_i(\mathbf{x}) \\ \text{subject to } & C(\mathbf{x}) = 0 \\ & \mathbf{D}\mathbf{x} + \mathbf{f} \geq 0. \end{aligned} \tag{3.60}$$

This corresponds to formulating control as a constrained quadratic program (QP), with weights determining each objective's influence (Section 6.2). This formulation has been used for balancing and motion capture tracking [Abe et al. 2007; da Silva et al. 2008b; da Silva et al. 2008a; Macchietto et al. 2009]. However, it can require substantial effort to properly tune parameters (e.g., [Jain et al. 2009]), and can lead to problems with objective terms that "fight."

To evaluate the benefits of our unconstrained prioritized optimization algorithm, we repeat the actuated chain (3.3.1) and squatting simulations (3.3.3) using a QP-based control. We reuse the same objectives, gains, and setpoints from these examples and combine them as described in

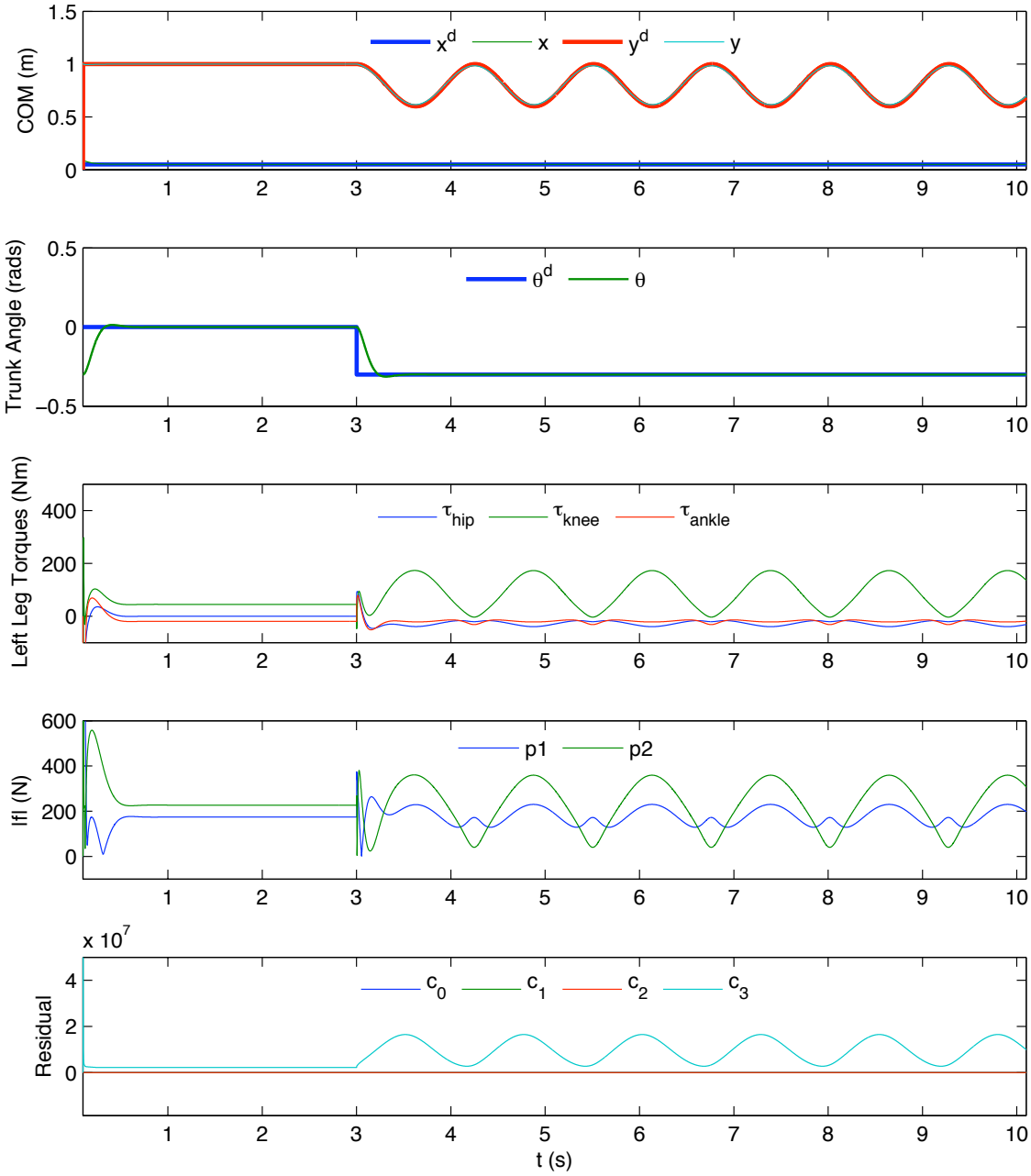

Figure 3.6: Planar squatting simulation results. From top to bottom, plots show: commanded vs. actual COM position, commanded vs. actual trunk orientation, left leg control torques, magnitude of bilateral contact forces for leg foot (p1 and p2 are the foot contact points), and residuals for all tasks specified to the prioritized solver (i.e., $E_i = || \mathbf{A}_i\mathbf{x} - \mathbf{b}_i ||$). Right leg joint details omitted for brevity.

(3.60). For the squatting simulation, we also include a regularization task (i.e., $\mathbf{A}_{reg} = \mathbf{I}$, $\mathbf{b}_{reg} = \mathbf{0}$, $\alpha_{reg} = 0.01$) to improve problem conditioning. Our implementation uses the interior-point QP solver [Boyd and Vandenberghe 2004] from [MOSEK].

For the actuated chain example, we were unable to find a set of weights for the QP formulation that achieved both tasks accurately. Instead, interference was observed between objectives in all attempted simulations. As one task was achieved, its contribution to the objective diminished, causing the optimization to favor the other objectives.

For the squatting example, many weight combinations cause the simulation to fail. With sufficient tuning, the QP-based controller results approach those of the prioritized solver. Figure 3.7 shows QP controller performance for a representative set of weight combinations. The QP-based control implementation runs at approximately half the speed of unconstrained prioritized optimization.

3.5 Summary

In this chapter, we introduced feature-based control. Feature-based control is an optimization-based approach for performing control in terms of high-level features of pose. Each feature is formulated as an objective. We describe four different objectives for: servoing to a set-point, reaching distant targets, regulating angular momentum, and minimizing user-specified torques.

Controllers are designed by combining multiple objectives in a prioritized manner. A prioritized optimization algorithm solves for joint torques satisfying objectives, subject to constraints. We validate this approach by presenting three examples.

In the following chapter we describe how to apply the objectives described in this chapter to more complex locomotion problems. We use a version of the prioritized optimization algorithm capable of dealing with unilateral constraints on actuation torques and contact forces. Because we target complex 3D characters, these problems must deal with balance; designing locomotion control requires time-varying actions to be synchronized.

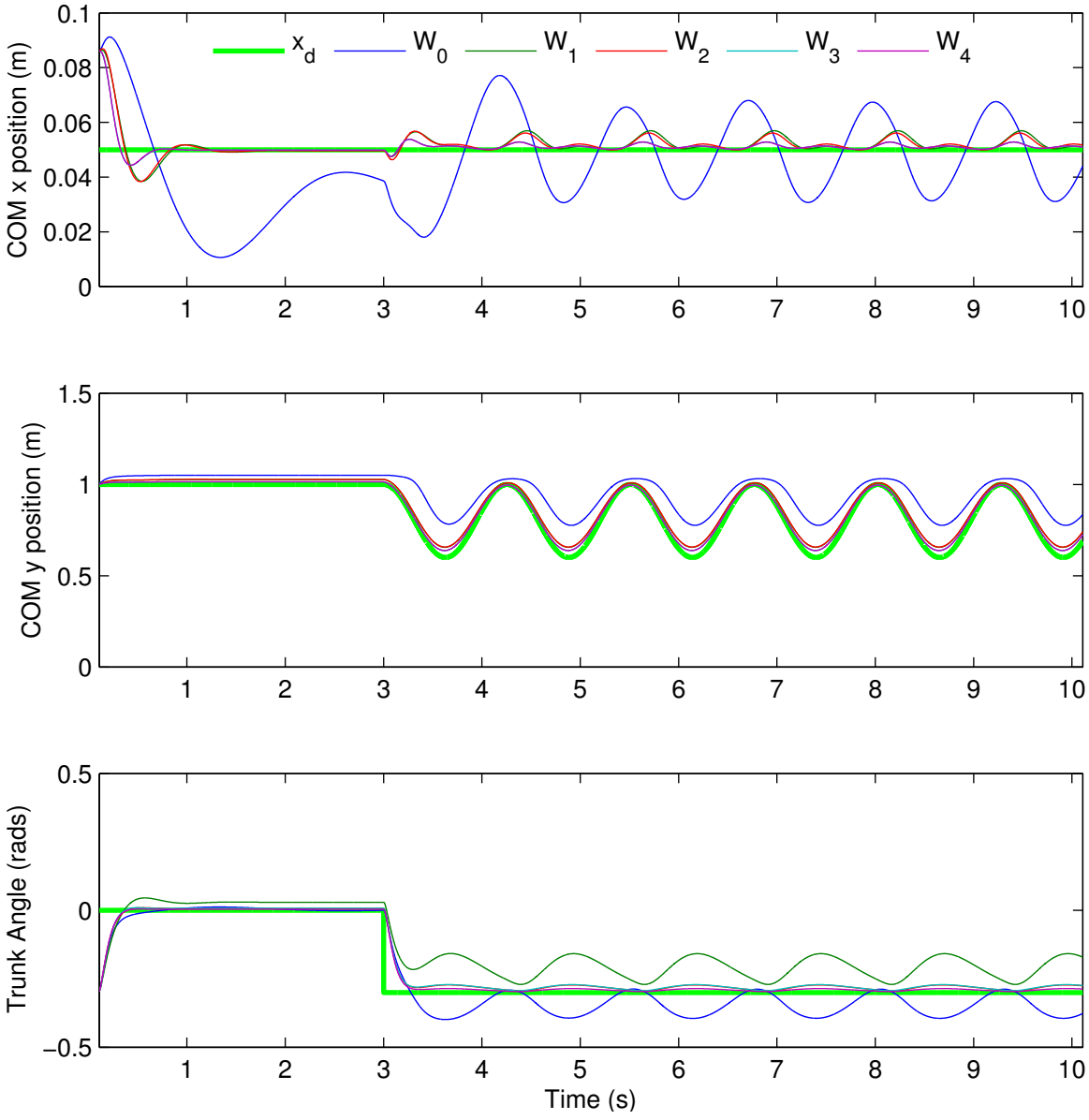

Figure 3.7: Weighted Multiobjective Optimization Squatting Results. Tracking of feature-space quantities for five different sets of weights is shown. Tasks weights $\mathbf{W}_i = (\alpha_1^i, \alpha_2^i, \alpha_3^i)$ are: $\mathbf{W}_0 = (10, 10, 0.1)$, $\mathbf{W}_1 = (50, 10, 0.1)$, $\mathbf{W}_2 = (50, 50, 0.1)$, $\mathbf{W}_3 = (100, 50, 0.1)$, $\mathbf{W}_4 = (100, 100, 0.1)$. As weights are increased, QP control output converges to prioritized solver output.

Chapter 4

Feature-Based Locomotion Control

In this chapter we apply feature-based control, introduced in Chapter 3, to locomotion tasks. We describe controllers for balancing, jumping, and walking. We show that locomotion control can be expressed in terms of a small, intuitive, set of features and objectives. Objectives can be interpreted in terms of balance and end-effector control. Control of multiple features is coordinated by a constrained prioritized optimization scheme that allows objectives to be resolved in priority order, while handling inequalities from limits on actuation torques and contact forces.

4.1 Balance

We begin with a basic balancing control task; this controller will later form the basis for our jumping and walking controllers. Our balancing strategies are based on previous results [Kudoh et al. 2006; Macchietto et al. 2009], with a number of improvements.

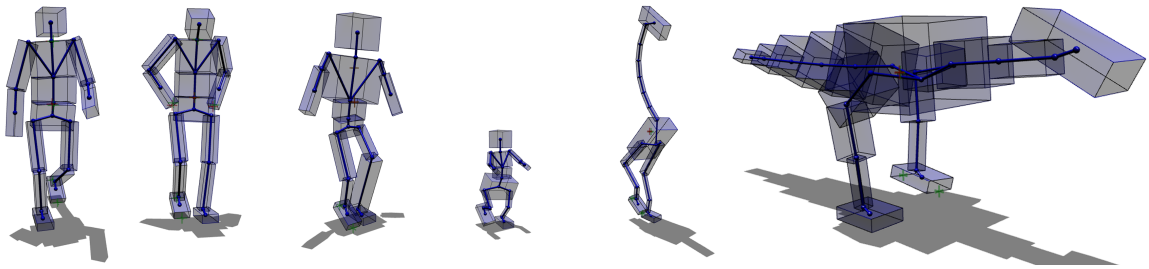

Figure 4.1: Feature-Based Locomotion Controllers. Controllers based on high-level features can be modified to create new styles, transferred to new characters and are robust to interactive changes in anthropometry. left to right: normal walking, "sad" walking, asymmetric character, "baby," ostrich, dinosaur.

Table 4.1: Objectives for Balance and Jumping. Each objective corresponds to the control of one feature.

Priority	Objective	Role
1	$E_{contact}$	Keep feet planted
2	$E_{com\parallel}$	Keep COM over base-of-support
	$E_{com\perp}$	Control COM height
3	E_{AM}	Minimize AM changes (Section 3.2.3)
	E_{pose}	Servo joints to a rest pose
	E_{ankle}	Servo ankles rel. to COM **(jump)**

Balancing involves several actions, summarized in Table 4.1. We divide actions into 3 priority levels, combining objectives at the same level with equal weight, (i.e., $E_1(\mathbf{x}) + E_2(\mathbf{x})$) (Section 6.2). With the exception of E_{AM} (Section 3.2.3), we use setpoint objectives for remaining features (Section 3.2.1).

The first priority level ensures the feet remain planted on the ground. This is enforced by an objective, $E_{contact}$, which keeps each foot's sole on the contact surface. This is done using a setpoint objective (Section 3.2.1), applied to each point on the bottom of the foot. The reference value $\mathbf{y}_r^{contact}$ is obtained by projecting each of these points onto the contact surface.

The second priority level is responsible for regulating linear momentum. To accomplish this, $E_{com\parallel}$ regulates the projection of the center-of-mass (COM) to the centroid of the base-of-support. Specifically, $\mathbf{y}_r^{com\parallel}$ is set to the average location of the contact points. Control of the COM component perpendicular to the contact surface uses a similar objective, $E_{com\perp}$; $y_r^{com\perp}$ is chosen based on the character's stature ($\approx 80\%$ max COM height). Varying $y_r^{com\perp}$ produces deep-knee squats.

At the lowest priority level, two objectives are used: an angular momentum (AM) objective E_{AM} compensates for unwanted rotations (Section 3.2.3), and a rest pose objective E_{pose} resolves remaining whole-body redundancies. The rest pose objective is a special case of the setpoint objective, which simplifies to:

$$E_{pose} = || \ \ddot{\mathbf{q}} - k_p(\mathbf{q}_r - \mathbf{q}) + k_v\dot{\mathbf{q}} \ ||^2 . \tag{4.1}$$

See da Silva et al. [2008b] for a discussion of how to compute position error (i.e., $\mathbf{q}_r^i - \mathbf{q}^i$) for spherical joints. We select a target for $\mathbf{y}_r^{pose} = \mathbf{q}_r$ that commands joints to a neutral pose; knees

are bent slightly to encourage rotation away from joint-limits. Pose feature gains are listed in Table 4.2; damping is set as described in Section 3.2.1. Contact constraints (Section 4.4.2) are also enforced for all foot contact points close to the ground.

Table 4.2: Pose Objective Gains.

Joint	k_p^{pose} (Nm)
Ankle	500
Shoulder	250
Elbow	50
Other	2000

The described balance controller is very stable across a broad range of parameters and characters (Section 4.5). Several whole-body strategies emerge automatically to achieve controller objectives. When applying large external perturbations affecting overall character posture, "windmilling" strategies emerge from AM regulation (Figure 4.2); arm "windmilling" is a common whole-body strategy used by people to avoid loss-of-balance, while standing [Kudoh et al. 2006; Macchietto et al. 2009]. The character also assumes natural postures coordinating diverse joint groups. Increasing reference COM height for $E_{com\perp}$ is first handled via knee extension; once knees reach limits, the character begins straightening his back and raising his arms. Commanding COM heights that exceed physical limits of the character eventually leads to failure, as no feasible solution exists.

4.2 Standing Jumps

We now describe a variety of jumping behaviors (Figure 4.4), created by simple modifications to the balance controller described above. A state machine determines reference values for individual balance controller features. Using the state machine, COM height is first lowered and then quickly raised, producing the jump. An additional objective, E_{ankle}, controls the ankle position in mid-air. Varying reference values yields different jumps, including twisting, in-air kicks/splits, and forward jumps.

The state machine for jumping includes the following stages: standing, compression, thrust, ascent, descent, and impact (Figure 4.3). During compress, we use the basic balance controller,

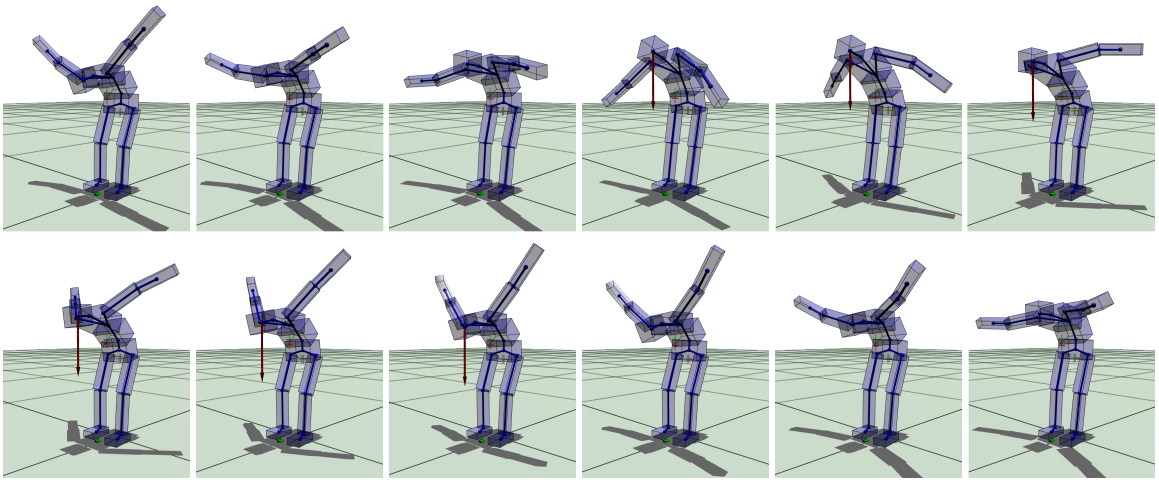

Figure 4.2: Arm "Windmilling" Compensates Disturbances: When applying cyclic perturbations (1000 N, duration: 0.1 s, interval: 0.1 s) to the standing controller, windmilling compensation strategies emerge automatically from AM control. Sequence order is left-to-right, top-to-bottom; elapsed time between images is 1/30 s

but lower the COM height, so that the character crouches. Then, thrust begins: $y_r^{com\perp}$ is quickly raised, while continuing to balance. When the COM passes a threshold height, we enter the ascent state. In this stage, COM tracking and contact objectives are disabled (i.e., $\alpha_{com} = \alpha_{contact} = 0$). Simultaneously, ankle control, E_{ankle} is enabled (i.e., $\alpha_{ankle} = 10$). This keeps the ankles at a fixed position relative to the COM, throughout flight, improving stability upon landing. During descent, and once the feet are about to touchdown, ankle rest pose is updated to match the ground slope. Upon impact, we disable E_{ankle} and re-enable E_{com} and $E_{contact}$ (i.e., $\alpha_{ankle} = 0$, $\alpha_{com} = \alpha_{contact} = 1$). The COM motion in this phase is determined by the balance controller, which helps decelerates the body. Once the COM starts moving upwards, a target objective (Section 3.2.2) $E_{com\perp}$ is used to bring the character back to a standing posture. All changes to COM height are done using a target objective. This avoids the need for careful trajectory design or gain tuning. For the standing jump, AM damping (i.e., $\mathbf{L}_r = \mathbf{0}$) is applied throughout the motion.

Jump with twist. To create twisting jumps, non-zero vertical AM is commanded about the COM during thrust. The greater the AM at take-off, the more the character will rotate in the air. Sample 90° and 180° twisting jumps are shown in the accompanying video.

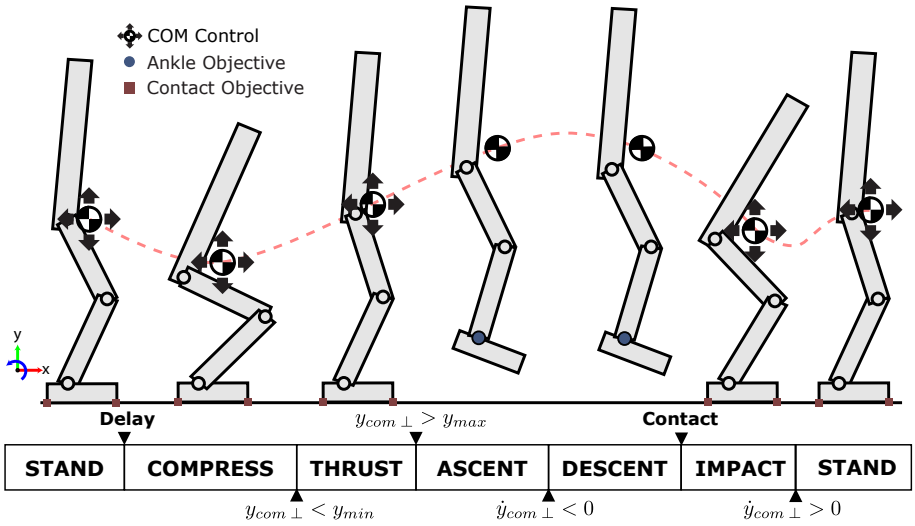

Figure 4.3: Planar illustration of jump controller actions/state machine. State transition are based on COM state and contact events

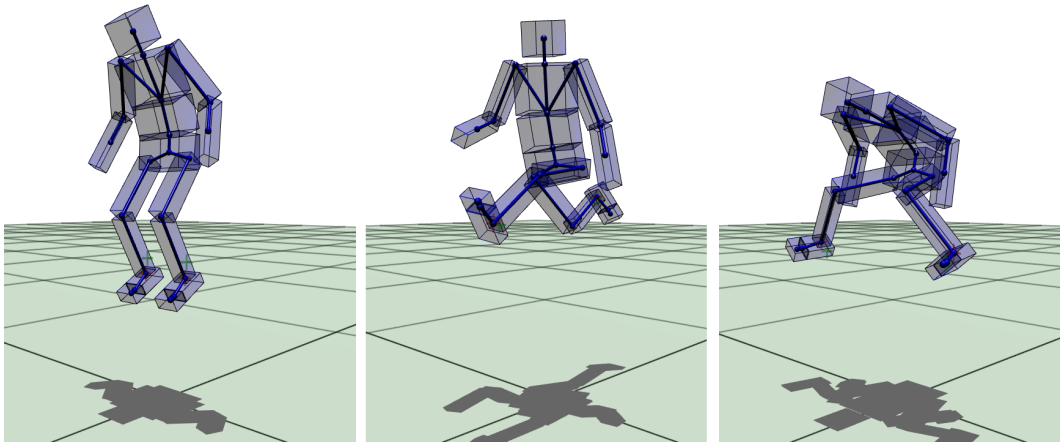

Figure 4.4: Sample jumping styles (left to right): Normal, Scissor-Kicks, Air-Splits

Jumping kicks and splits. To create scissor kick or air-split motions (Figure 4.4), standing-jump controller values for $\mathbf{y}_r^{ankle}$ are varied throughout flight. To do this, we rotate the ankle offset relative to the COM about a user-specified axis, reversing the direction of motion once descent begins. Accompanying videos show natural counter-rotation of the torso during ballistic motion.

Forward Jumps. Jumps that produce large COM displacements can be created by further extending the acrobatic controllers (Figure 4.5). During thrust, a small displacement in COM position is commanded in the direction of desired travel. During flight, ankle position trajectories are modified to prepare for landing. See the accompanying video for more jumping

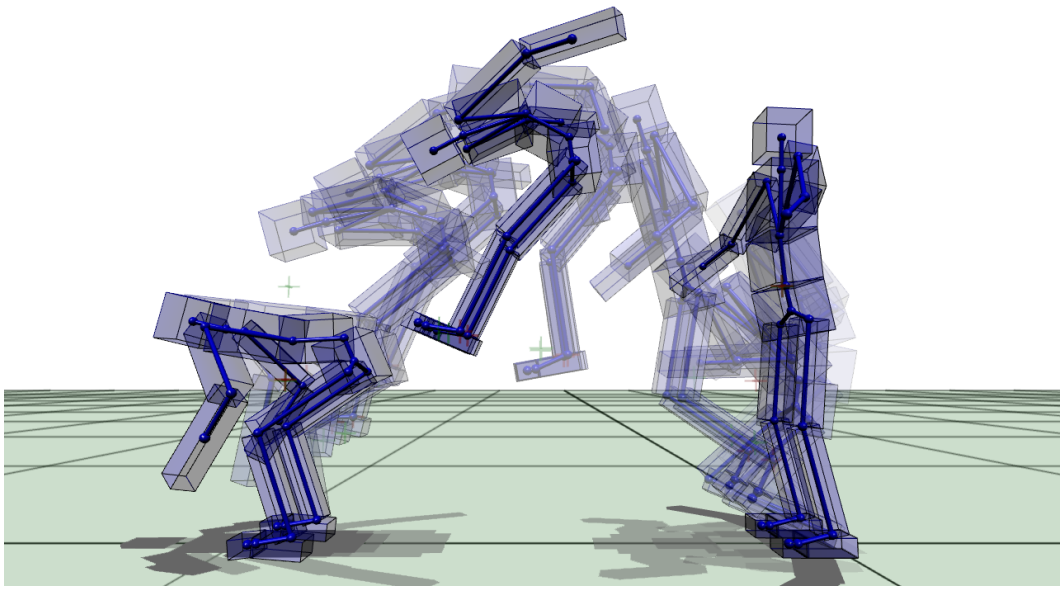

Figure 4.5: Sample Forward Jump: Modifying in-place jump controller yield forward jumps

examples.

4.3 Walking

We now describe a locomotion controller capable of walking at different speeds, with different stepping heights, and in a variety of styles. The features used by this controller can be divided into two categories. First, foot and COM features are controlled to drive the character forward. A state machine governs targets for these features. The remaining objectives are used to maintain balance and control additional nuances of style. These objectives are independent of the state machine; many of them are derived from the balancing controller in Section 4.1. The complete list of features/objectives is summarized in Table 4.3.

4.3.1 State-dependent objectives

Control of the feet and the COM is governed by a state machine. Each foot can be in one of four states: swing, plant, support, and heeloff. State transitions are determined as shown (Figure 4.6). At specific state machine transitions, targets for the feet and COM are recomputed based on user-specified properties of gait, such as step length and duration.

Foot control. Two objectives are used to control the feet, $E_{contact}$ and E_{swing}. When a foot is in one of the contact states (plant, support, heeloff), the $E_{contact}$ objective enforces contact conditions, and is implemented as in the balancing controller. Depending on the current state, either the front, back, or all four corners of the foot are kept in contact with the ground (Figure 4.6).

When a foot is in the swing phase, the E_{swing} target objective controls its trajectory. This objective first raises the foot to a user-specified step-height h and then lowers it, all the while moving the foot forward. One target objective is used for the toe, and one for the heel. The desired swing phase duration T is set by the user (Table 4.3).

More specifically, the target location $\mathbf{y}_r^{swing}$ for the heel is split into a component parallel to the ground plane $\mathbf{y}_r^{swing\,\parallel}$ and a target height $y_r^{swing\perp}$. The parallel component is determined when the foot enters the swing phase, as an offset from $\mathbf{y}^{stance}$, the heel's position at the start of the swing phase, as:

$$\mathbf{y}_r^{swing\,\parallel} = \mathbf{y}^{stance} - 2\,\alpha\,l_{hip}\,\mathbf{d}_\perp + l_{step}\,\mathbf{d}_\parallel \tag{4.2}$$

where $\mathbf{d}_\parallel$ is the direction of travel, $\mathbf{d}_\perp$ is the right-handed direction perpendicular to $\mathbf{d}_\parallel$, and l_{hip} is the stride width. The indicator α is $+1/-1$ for the right and left swing foot, respectively. For the first half of the swing phase duration, $y_r^{swing\perp}$ is set to a user-specified step-height h. For the remaining duration, it is set to zero. The target objective for the toe is identical to that of the heel, but delayed by a small amount.

Note that the swing foot objectives do not precisely specify foot motion. Foot motion can deviate for many reasons, including conflicts with higher-priority tasks, unmodeled disturbances, and inconsistencies between commands and the character's foot size. Objectives are meant to "guide" the feet in a sufficient way to avoid toe-stubbing, while moving the foot to the next foot plant.

COM control. The COM is controlled along a sinusoidal path that propels the character forward, placing the COM over the stance foot for balance. Whenever a swing foot enters the plant phase, this target trajectory is recomputed. To synchronize the COM and the feet, the desired COM trajectory duration is set to the actual duration of the previous swing phase, $T_{com}^i = T_{swing}^{i-1}$. This simple feedback mechanism encourages the character to enter a stable limit-cycle after only a few steps. The duration of the COM trajectory is parameterized with a

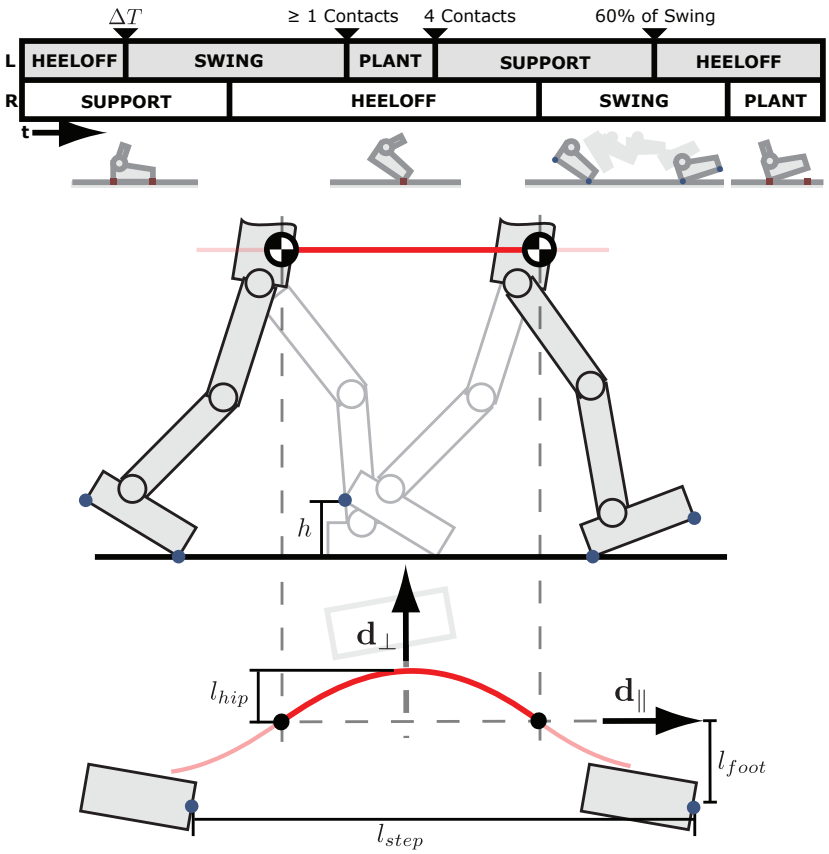

Figure 4.6: Walking Actions and Parameters. Top: A state machine is used to coordinate contact ($E_{contact}$), COM ($E_{com\|}$), and stepping (E_{swing}) objectives. **Middle/Bottom:** Illustration of parameters (Table 4.3) used to specify swing targets $\mathbf{y}_r^{swing}$ and COM trajectory $\mathbf{y}_r^{com\|}$.

phase variable $\phi \in [0...1]$.

COM motion parallel to the ground plane is commanded along a trajectory:

$$\mathbf{y}_r^{com\|}(\phi) = \phi \mathbf{p}_1 + (1-\phi)\mathbf{p}_0 + \alpha l_{hip} \sin(\pi\phi)\mathbf{d}_\perp \tag{4.3}$$

relative to the initial COM position $\mathbf{p}_0$, with:

$$\mathbf{p}_1 = \mathbf{y}^{stance} - \alpha\, l_{foot}\, \mathbf{d}_\perp + \frac{1}{2}\, l_{step}\, \mathbf{d}_\| \tag{4.4}$$

where l_{hip} and l_{step} specify hip shift and step length respectively (Table 4.3). In our walking controller, we leave the COM motion perpendicular to the ground plane unconstrained. This allows the walking motion to pivot on the stance leg, rather than walking with some predefined height or attempting to hand-craft trajectories that may conflict with the system dynamics. To

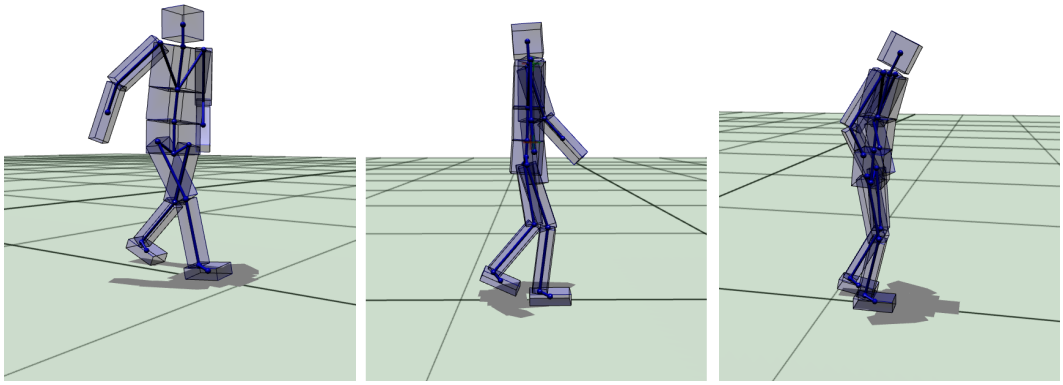

Figure 4.7: Sample walking styles (left to right): Fast, slow, "sad"

adjust walking height, we vary the knee angle in E_{pose}.

4.3.2 State-independent objectives

The remaining objectives do not depend on the state machine. As with in-place motions, we regulate AM to improve balance (Section 3.2.3). Biomechanical studies have indicated that AM is tightly regulated during human walking [Herr and Popović 2008]. This quantity has also proven useful in control optimization [Wang et al. 2009]. To our knowledge, ours is the first method which actively regulates AM for human locomotion.

For style, an additional objective E_{arms} is also included to encourage passive arm motion. E_{arms} uses the minimum torque objective (Section 3.2.4), selecting shoulder and elbow joints via the matrix $\mathbf{S}$. The combination of AM damping and minimum upper-body torque control produces natural-looking in-phase arm swing and hip-torso counter oscillation.

Two additional objectives are used. To control the character's posture, we use a setpoint objective $E_{trunk\|}$ specifying the location of the base of the neck relative to the COM:

$$\mathbf{y}_r^{trunk\,\|} = \mathbf{y}_r^{com\|} + \mathbf{d}_{\|}\, l_{trunk}. \tag{4.5}$$

Lastly, a setpoint objective E_{head} is used to stabilize the orientation of the head [Pozzo et al. 1990]. A quaternion setpoint $\mathbf{q}_{head}$ specifies the desired orientation of the head about $\mathbf{d}_{\perp}$, while also pointing the head in the direction of travel.

Table 4.3: Objectives and Parameters for Walking. Top: Objectives uses for Walking **Bottom:** Walking controller parameters. See Figure 4.6 for an illustration of these quantities. Quantities in parameter table are commanded values. These will differ from actual values.

Priority	Objective	Task
1	$E_{contact}$	Keep stance foot planted
2	$E_{com\parallel}$	Move COM parallel to ground plane
	$E_{trunk\parallel}$	Control neck position wrt COM
3	E_{swing}	Control swing foot
	E_{AM}	Dampen angular momentum
	E_{pose}	Servo joints to rest pose
	E_{arms}	Minimize arm torques
	E_{head}	Servo global orientation of head

Parameter	Symbol	Value
Swing Duration	T	0.3 - 0.7 s
Step Length	l_{step}	0 - 0.7 m
Swing Height	h	0 - 0.1 m
Foot Spread	l_{foot}	0 - 0.03 m
Swing Delay	ΔT	0 - 0.3 s
Hip Shift	l_{hip}	0 - 0.3 m
COM offset	l_{trunk}	± 0.3 m
Head Orientation	$\mathbf{q}_{head}$	$\pm 180°$

4.4 Prioritized Optimization

We next describe details of the optimization solved to enforced objectives for balancing, jumping, and walking.

At each instant of a simulation, the controller determines the following vector of unknowns:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{4.6}$$

where $\boldsymbol{\tau}$ denotes joint torques, $\ddot{\mathbf{q}}$ denotes joint accelerations, and $\boldsymbol{\lambda}$ denotes weights in the linearized friction cone bases. To ensure that generate contact forces are not sticky we apply con-

strained prioritized optimization algorithm (Section 6.3).

4.4.1 Equality Constraints

As shown in Equation 3.55, the equations of motion:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{h} = \boldsymbol{\tau} + \mathbf{J}_c^T \mathbf{f}_c \tag{4.7}$$

relate joint accelerations $\ddot{\mathbf{q}}$, joint torques $\boldsymbol{\tau}$, and contact forces $\mathbf{f}_c$. The quantities $\mathbf{M}$ and $\mathbf{h}$ are the joint space inertia matrix, Coriolis/centrifugal and gravitational forces respectively. The contact force Jacobian $\mathbf{J}_c$ maps generalized velocities $\dot{\mathbf{q}}$ to world space Cartesian velocities at contact points. Contact forces:

$$\mathbf{f}_c = \mathbf{V}\boldsymbol{\lambda} \tag{4.8}$$

are expressed in the basis of the linearized friction cone $\mathbf{V}$ [Abe et al. 2007]. Substituting (4.8) and (4.6) into (4.7), and rearranging gives the equality constraints:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{M} & \mathbf{J}_c^T\mathbf{V} \end{bmatrix} \mathbf{x} - \mathbf{h} = \mathbf{0}. \tag{4.9}$$

4.4.2 Inequality Constraints

To accurately solve for control torques, it is necessary to account for contact forces arising from joint-limits and ground interaction. Together, these define the inequality constraints:

$$\mathbf{D}\mathbf{x} + \mathbf{f} \geq \mathbf{0} \tag{4.10}$$

used by our simulator.

Ground Contact. Ground contact forces must be strictly repulsive and respect friction [Abe et al. 2007; da Silva et al. 2008b; Fang and Pollard 2003]. In our experience, strictly enforcing zero acceleration at contact points [Abe et al. 2007] leads to numerically sensitive/infeasible problems. We have found that using non-penetration constraints [Baraff 1994], expressed in the linearized friction cone basis, produces more stable results. These conditions can be summarized

by the inequality constraints:

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{4.11}$$

$$\mathbf{a}_c \equiv \mathbf{V}^T \mathbf{J}_c \ddot{\mathbf{q}} + \mathbf{V}^T \dot{\mathbf{J}}_c \dot{\mathbf{q}} + \dot{\mathbf{V}}^T \mathbf{J}_c \dot{\mathbf{q}} \geq \mathbf{0} \tag{4.12}$$

Although complementarity constraints are not enforced (i.e., $\boldsymbol{\lambda}^T \mathbf{a}_c = \mathbf{0}$), we have not found this simplification to adversely impact controller performance or stability.

Joint Limits. To prevent the character from assuming unnatural postures, we include additional constraints in the contact Jacobian:

$$\mathbf{J}'_c = \begin{bmatrix} \mathbf{J}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{lim} \end{bmatrix} \tag{4.13}$$

for all joints at limits. For a joint, whose generalized coordinate configuration $\mathbf{q}^i$, has range $\mathbf{q}^i_l \leq \mathbf{q}^i \leq \mathbf{q}^i_u$, we set the rows of $\mathbf{J}_{lim}$ to the standard basis $\mathbf{e}_i$ when $\mathbf{q}^i = \mathbf{q}_l$ and to $-\mathbf{e}_i$ when $\mathbf{q}^i = \mathbf{q}_u$. This increases the number of $\boldsymbol{\lambda}$s in $\mathbf{x}$ (Equation 4.6).

Torque Limits. Bounds on torques help prevent superhuman feats of strength and are expressed as: $-\boldsymbol{\tau}^i_{max} \leq \boldsymbol{\tau}^i \leq \boldsymbol{\tau}^i_{max}$.

We find that directly including the equality constraints $C(\mathbf{x}) = \mathbf{0}$ (Equation 4.9) as constraints in the QP optimization can lead to infeasible solutions due to numerical issues, particular at contact transitions where rows of $\mathbf{J}_c$ become linearly dependent to machine precision. Instead, we obtain stable results by adding a top-priority objective $E(\mathbf{x}) = || \, C(\mathbf{x}) \, ||^2$. Since this objective has higher priority than all other objectives, it will always be minimized to zero, ensuring that the equality constraints are satisfied.

4.5 Results

We now describe results of applying feature-based controller design to balancing, standing jump, and walking motions, seen in Figures 4.2, 4.4, 4.5, 4.7, and the accompanying video.

Simulation Details. Our simulator uses Featherstone's algorithm, and provides automatic computation of Jacobians and equations-of-motion [Featherstone 2008]. Equations are integrated using the semi-implicit scheme of Guendelman et al. [2003]. Ground contact is modelled, using an inelastic impulse-based model, that includes friction ($\mu = 1$). Our double-precision single-threaded implementation runs at $50 - 100\%$ real-time, on a Dual Core 3GHz Intel Xeon CPU with 4GB RAM, running OSX 10.5, for all presented examples. Speed varies depending on character complexity, number of active contacts, number of prioritization levels, and the choice of QP solver. Our default character model has 35 degrees of freedom, with height and weight corresponding to a 50th percentile North American male. Skeletal dimensions and link mass are taken from Winter [2004]. Link inertias are calculated using uniform density shapes scaled to match skeletal dimensions.

Once optimal torques are computed, the joint accelerations $\ddot{\mathbf{q}}$ can be computed in two ways. First, one may directly apply forward dynamics using the torques. Alternatively, because the optimizer outputs $\ddot{\mathbf{q}}$ values as well as torques (i.e., both are included in $\mathbf{x}$), one may instead directly use the output of the optimizer. Because complementarity is not enforced during optimization, using these $\ddot{\mathbf{q}}$'s may lead to slight numerical inaccuracies. Furthermore, one must use a QP solver that guarantees that all constraints will be satisfied: interior-point methods do satisfy all constraints, whereas active set methods may not, leading to small sticky (bilateral) forces. However, using the optimizer output allows dynamics to be advanced at significantly larger timesteps. Specifically, using forward dynamics integration and control must be done at 1 kHz to maintain stability; the simulation can operate at 100Hz using $\ddot{\mathbf{q}}$ from (4.6). In our tests, we developed the controllers using $\ddot{\mathbf{q}}$ values from the optimizer, and then perform the final simulations using forward dynamics.

Balance & Jumping. Starting from the balance controller, it was straightforward to produce jumps of different heights, for characters with varying skeletal properties, requiring only minor changes to parameters controlling hopping height. Despite large changes in motion, the controller produced consistent landings. This robustness is atypical of joint-space parameterization. The examples we demonstrate are qualitatively similar to those of Wooten [1998], who explored a vast array of acrobatic maneuvers for simulated humans.

Starting from the basic jump controller, it took about 30 minutes to make the character perform both twisting jumps and in-air acrobatics. The feature-based parameterization also provides a

useful dimensionality reduction. Only 3 parameters (i.e., crouch depth, thrust phase duration, and reference AM) vary between standing and twisting jumps. Designing similar motions using conventional methods would have been very difficult, requiring careful trajectory design to avoid unwanted rotations prior to flight. Injecting AM in a coordinated manner would also be very fragile and time consuming to design.

Walking. Using the feature-based representation, we generate walking in a variety of styles. This can be done by reasoning about the desired look and feel of the motion. For example, to generate the sad walk example shown in our video, we begin with the slow walk, activate a set-point objective to place the character's hands in his "pockets," tilt the head forward, add a little bit of forward lean to the body, and add a small amount of knee bend to the rest pose. This can be done without concern for coupling between different features. For example, since we are explicitly controlling the COM in our walking controller, trunk lean angle does not directly influence forward speed. This allows lean angle to be varied over a large range (Table 4.3), without worrying about its effect on stability. This is not the case with many joint-space walkers, which tightly couple lean and forward velocity (e.g., [Yin et al. 2007]). Furthermore, our method allows interactive changes to gait parameters, by directly adjusting control parameters such as those in Table 4.3.

Changes to Body Shape. Generalization of motion synthesis methods to new characters is an important goal. This has previously been demonstrated for kinematic methods [Hecker et al. 2008], and to a lesser extent for physics-based systems [Coros et al. 2009; Hodgins and Pollard 1997; Tsai et al. 2010; Wang et al. 2009]. Model-based approaches [Abe et al. 2007; Coros et al. 2010; Macchietto et al. 2009], such as ours, significantly expand control generalization capabilities.

Our controllers are robust to large changes to skeletal and inertial properties. We demonstrate this for both balancing and walking by changing morphology for our biped during simulation (Figure 4.1). No control parameters are changed for these experiments. Limb dimensions are increased by a factor of 2, with mass and inertia scaled appropriately.

Furthermore, we can apply the same controllers to characters with vastly different topologies and mass distributions (Figure 4.1). Aside from rest pose, all of our features are independent of character topology. We parameterize all of our characters in such a way that 0 can be used

for the rest pose for all joints. Our implementation automatically computes all necessary features, Jacobians, and objective terms without requiring any additional effort. We find controllers can be successfully applied onto new characters *without any manual modification to the existing controller whatsoever*. This illustrates the power of the feature-based representation as an abstraction for reasoning about and creating motion.

Benefits of Prioritization. In our experiments, we have found prioritized optimization to have several advantages over weighted multiobjective combination. Although weighted optimization is sufficient for control with a small set of objectives [Abe et al. 2007; Jain et al. 2009], it can behave poorly when many objectives are enabled/disabled at runtime. For example, despite only minor differences between the balance and jumping controllers, we were not able to generate jumping motions using weighted optimization. Results were numerically sensitive, producing infeasible problems, despite activating only one additional objective (e.g., E_{ankle}). As controllers become more complex and use a larger set of objectives, tuning objective weights becomes more difficult and time-consuming. From a design perspective, using prioritization allows control to be layered, improving workflow. Control of fundamental motion features can be "locked-in", without worrying about impact of lower-priority objectives on stability or motion execution.

In our experiments with in-place standing motions, we found prioritized optimization to be more robust to external disturbances. Because weighted optimization does not explicitly decouple objectives, disturbances affect all objectives. As drift increases in one objective, weighted optimization will begin to favor other objectives (e.g., error in E_{com} will increase, while E_{pose} decreases), eventually leading to task failure. Prioritized optimization does not suffer from this failure mode. Instead, tracking of low-priority features is sacrificed to minimize errors in higher priority goals. This yields human-like disturbance rejection, consistent with hypotheses from motor neuroscience [Todorov and Jordan 2002]. See the project web page for a comparison of weighted/prioritized optimization.

We also converted our walking controller to a weighted optimization. However, minimizing interference between the different objectives was extremely difficult and sensitive to parameter tuning. Stiff-looking walking can be obtained by disabling all objectives except for $E_{contact}$, $E_{com\parallel}$, E_{swing}, and E_{pose}. Stylistic terms such as E_{AM} and E_{arms} proved especially sensitive and required careful tuning. Small changes in these objectives produced motion with "flailing"

arms or excessive torso-counter rotation that "fought" rest-pose objectives. Coupling between user control parameters also increased. Because of these issues, we do not believe we would have been able to develop a good set of objectives in the first place without using prioritized optimization.

Prioritized optimization has a few disadvantages as compared to weighted optimization. Prioritized optimization is slower and more work to implement, since each priority level requires solving both a QP and a SVD. For locomotion control, prioritized optimization requires the use of double-precision arithmetic for numerical stability, whereas single-precision arithmetic is sufficient for weighted optimization. In our attempts to use prioritized optimization with single-precision arithmetic we found that Jacobians for high priority objectives (e.g., $E_{contact}$), can become rank deficient to machine precision. This makes prioritized optimization sensitive to preconditioning techniques used by the QP solver. Weighted multiobjective combination does not have these problems, since adding many low-rank matrices together can produce full-rank ones. When the problem is formulated without unilateral constraints these issues can be avoided (Section 6.1).

4.6 Summary

In this chapter, we presented a set of locomotion controllers for complex physics-based characters based on controlling high-level motion features. Features are resolved in prioritized order subject to equality constraints, defined by dynamics, and unilateral constraints, from contact force, actuation, and range of motion limits. By exploiting prioritization, our method reduces objective weight tuning, increases robustness, and allows controllers to be designed incrementally, while decreasing coupling between control objectives.

Although our controllers only use a small number of features, many natural behavior properties arise, including human-like arm movement and hip-swaying in walking. These motion features cannot easily be generated using previous joint-space approaches.

To synchronize time-varying feature targets, we propose a number of action-specific state machines. For walking, foot plants are determined using a simple strategy with minimal feedback. This approach has some limitations. Runtime updates to hip shift, swing duration, and step length must be coordinated to ensure the COM moves sufficiently close to the stance foot, to

allow the stance leg to support the body. As swing duration decreases motion becomes more stable since the walker spends less time in single-support. Furthermore, because these curves are defined in world coordinates, the controller is not robust to perturbations.

To address these issues, Chapter 5 proposes a strategy for selecting optimal feature commands, by planning in the low-dimensional feature space, while simultaneously considering dynamics and terrain characteristics.

Chapter 5

Robust Physics-Based Locomotion Using Low-Dimensional Planning

> In nature, unlike in the laboratory, straight-line, steady-speed locomotion is the exception rather than the rule.
>
> [Dickinson et al. 2000]

Generating robust locomotion control of physically-simulated characters is very challenging. At every instant, control must select forces that affect not only immediate state, but also future motion completion. This is especially important when moving across irregular terrain with little margin for error. In these situations, foot step locations and modes of travel must be jointly selected to set up the next step. For example, a character walking on flat ground may need to switch to a different gait, such as running or jumping, when traversing stepping stones or

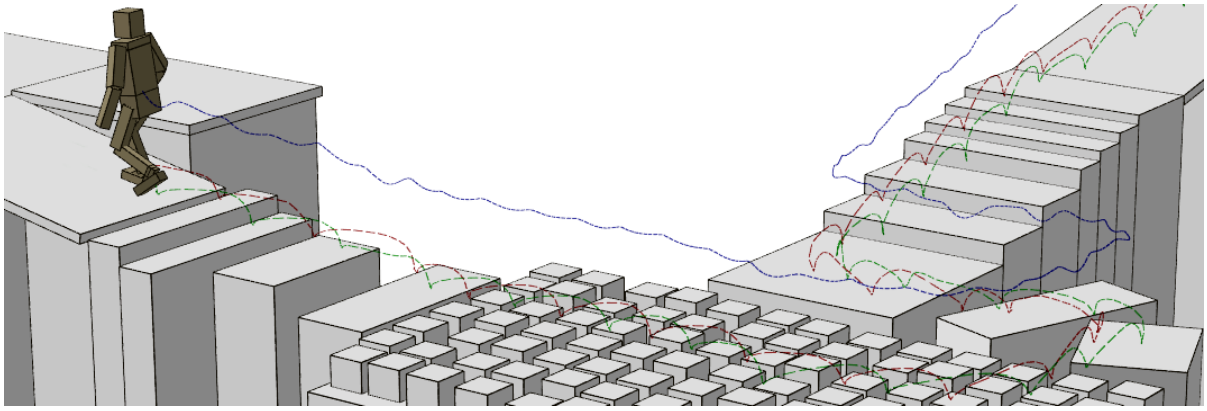

Figure 5.1: Interactive locomotion control over varied terrain. Gait, footsteps, and transitions are automatically generated, based on user-specified goals, such as direction, step length, and step duration. In the above example, a user steers the biped across uneven terrain with gaps, steps, and inclines.

chasms.

Planning these types of motions remains an open problem. Performing a full spacetime optimization of the motion is extremely expensive, and cannot handle unforeseen disturbances, changes to the environment, or new user-specified goals. Feature-based control methods, described in previous chapters, provide a number of useful abstractions for designing controllers. However, complex actions still require task-specific state-machines and trajectories. As an alternative, simplified dynamics models that enable online planning (of feature actions) can be used. However, existing models are gait specific and do not plan control over multiple foot-steps.

This chapter introduces a locomotion controller for full-body 3D physics-based characters that plans joint torques by optimizing a low-dimensional physical model. At each instant, the controller optimizes a plan over multiple phases of locomotion. To generate the plan, full-body dynamics are approximated by a set of closed-form equations-of-motion, allowing replanning to occur at every time step. The controller then chooses optimal full-body torques according to the current plan, while maintaining balance. Depending on the task goals and terrain, different gaits, including walking, running, and jumping, and gait transitions, emerge naturally. When navigating over terrain with a limited number of footholds, optimal paths are automatically determined. We demonstrate this technique on several uneven terrain navigation examples, including gaps, inclines, stairs, large drops, and stepping stones. Robustness to external disturbances and projectile avoidance capabilities are also shown. Our method requires no preprocessing and no motion capture data; a small set of user-level parameters allows control over the final motion.

5.1 Overview

We describe a locomotion controller that calculates joint torques $\boldsymbol{\tau}$ for a physics-based character (Figure 5.2). At each instant of the simulation, a plan $\mathbf{S}^{\star}$ is computed by nonlinear optimization of the motion over the subsequent two footsteps. Since trajectory optimization [Witkin and Kass 1988] for full-body locomotion remains very computationally expensive [Liu et al. 2005; Wampler and Popović 2009], we optimize the trajectory of a low-dimensional preview model instead. Full-body torques are then chosen such that the full-character's COM follows instantaneous plan accelerations.

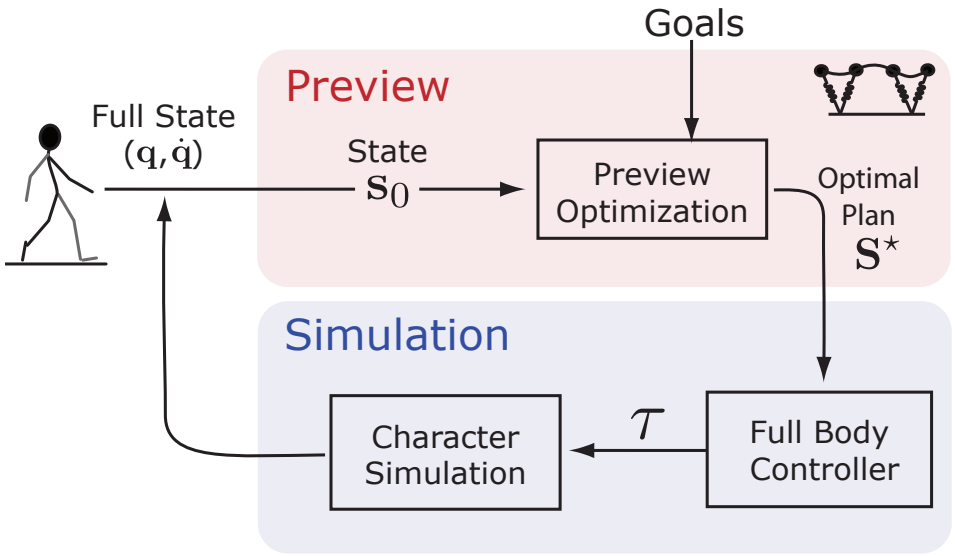

Figure 5.2: System Overview. At each time step, character state $\mathbf{q}, \dot{\mathbf{q}}$ is mapped to a lower-dimensional preview model state, $\mathbf{s}_0$. Next, a preview optimization generates an optimal plan $\mathbf{S}^\star$ that meets user-specified goals. Lastly, the full-body controller optimization calculates joint torques $\boldsymbol{\tau}$, based on the instantaneous accelerations from $\mathbf{S}^\star$. Torques are applied to the simulation, resulting in a new character state.

The preview model is based on the Spring-Loaded Inverted Pendulum [Full and Koditschek 1999] (SLIP) (Section 5.2). A fixed preview schedule divides the next two footsteps into 6 phases: double stance, left single-stance, flight, double stance, right single-stance, and flight. The optimization includes objectives to achieve target goals (e.g., desired step length and heading) and objectives to compensate for simplifications (e.g., foot position relative to hip). The optimization produces different gaits by omitting certain phases: omitting double-stance leads to jogging, while omitting flight leads to walking. The plan is computed by optimizing control parameters $\mathbf{U}$, for the SLIP motion $\mathbf{S}$ during these 6 phases. We make simplifying approximations to derive closed-form SLIP equations that allow forward dynamic simulation without numerical integration. The optimization yields an optimal plan $\mathbf{S}^\star$.

Using the optimal plan $\mathbf{S}^\star$, $\boldsymbol{\tau}$ is determined by a second optimization (Section 5.4). This optimization attempts to have the full character match the instantaneous accelerations of $\ddot{\mathbf{S}}^\star$, while also maintaining balance. Given torques $\boldsymbol{\tau}$, a simulator computes joint accelerations $\ddot{\mathbf{q}}$ and contact forces $\mathbf{f}_c$, and updates the full-character state $(\mathbf{q}, \dot{\mathbf{q}})$ by integration. This process of replanning and torque optimization is repeated at every time-step. Instead of replanning every simulation timestep, faster performance can also be obtained by replanning less frequently.

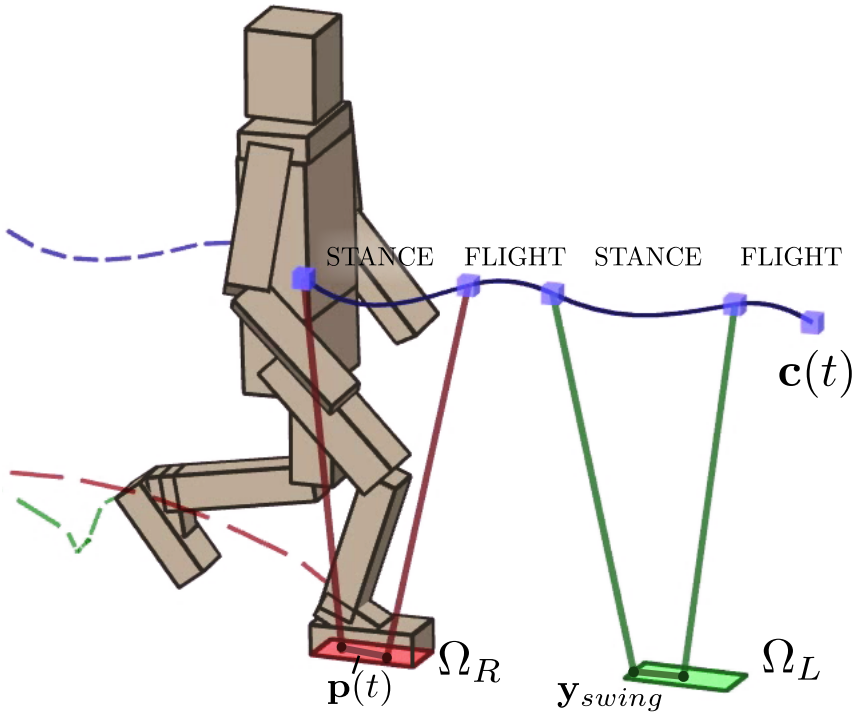

Figure 5.3: Low-dimensional planning. At each instant, a low-dimensional plan $\mathbf{S}^\star$ is computed, specifying COM ($\mathbf{c}(t)$) and COP ($\mathbf{p}(t)$) motion, as well as the next foot plant ($\mathbf{y}_{swing}$). Motion is divided into stance and flight phases, with COM motion modeled as described in Sections 5.2.2 and 5.2.3. COP motion uses a linear model (Equation 5.3). A running motion is shown above, with footplants Ω_R/Ω_L.

5.2 Preview Simulation

This section describes the low-dimensional model used to simulate dynamics over multiple phases of motion. Stance phases are represented with a modified SLIP model, with simplifications that allow closed-form integration. The resulting motion over multiple phases is piecewise polynomial, allowing analytic integration of objective function terms (Section 5.3). Ballistic phases are represented by COM motion only. Initial conditions for the preview simulation come from the current full-body character state, which provides initial COM position $\mathbf{c}_0$ and velocity $\dot{\mathbf{c}}_0$, initial heading α_0 and angular velocity $\dot{\alpha}_0$, and right/left foot contact polygons Ω_R/Ω_L. α_0 is set to the orientation of the pelvis about the vertical axis.

5.2.1 Stance Model

When the character has one or more feet on the ground, a SLIP model [Full and Koditschek 1999] is used to describe character motion. We modify the SLIP in several ways. First, we add an

explicit representation of the contact polygon corresponding to the convex hull of the support feet. Second, since exact SLIP equations must be numerically integrated, we use an approximate model that decouples COM motion in directions parallel and perpendicular to the ground plane. This allows equations of motion and objective functions to be computed in closed-form. Lastly, rather than assuming a fixed COP position, we allow it to move within the support polygon.

The SLIP model is parameterized by a COM position $\mathbf{c}$ and a heading α, representing orientation of the pelvis about the vertical axis. At any given instant, the complete degrees-of-freedom of the model are:

$$\mathbf{s}_s(t) = \begin{bmatrix} \mathbf{c}(t) \\ \dot{\mathbf{c}}(t) \\ \alpha(t) \\ \dot{\alpha}(t) \\ \Omega_R \\ \Omega_L \end{bmatrix} \tag{5.1}$$

where Ω_R/Ω_L are the support regions for the right/left foot respectively. The support regions are constant throughout a given phase. When a foot is in the air, its support region is empty ($\Omega = \emptyset$). Otherwise, its support region is modeled as a quadrilateral determined by projecting the sole of the foot onto the contact surface (Figure 5.3)

The complete set of single and double stance control parameters used by our model, are summarized by an 8-dimensional vector:

$$\mathbf{u}_s = \begin{bmatrix} T \\ \mathbf{p}_0 \\ \mathbf{p}_T \\ r_0 \\ r_T \\ \ddot{\alpha} \end{bmatrix}. \tag{5.2}$$

The duration of the phase is set by the parameter T. The SLIP model is controlled by a linear spring that connects the COM, $\mathbf{c}$, to COP, $\mathbf{p}$, on the ground. The spring has rest length $r(t)$ and a fixed stiffness k. Rather than fixing the COP as is typically done (e.g., [Kajita et al. 2001]), we allow a time-varying COP. Walking with a fixed COP corresponds to walking on pin-like feet

and does not take advantage of the foot's whole support region. In humans, foot rolling is an important factor in the production of smooth, energy-efficient walking [Adamczyk et al. 2006]. Both the rest length and COP are controlled linearly:

$$\mathbf{p}(t) = \frac{t}{T}(\mathbf{p}_T - \mathbf{p}_0) + \mathbf{p}_0 \tag{5.3}$$

$$r(t) = \frac{t}{T}(r_T - r_0) + r_0 \tag{5.4}$$

where $\mathbf{p}_0/\mathbf{p}_T$ are the start/end COP locations, and r_0/r_T are the start/end rest lengths. Angular motion is controlled by setting a constant angular acceleration $\ddot{\alpha}$.

5.2.2 Stance Dynamics and Simulation

We now describe SLIP equations-of-motion used to obtain $\mathbf{s}(t)$ (Equation 5.1), given initial conditions $\mathbf{s}_0$ and control $\mathbf{u}_s$. As discussed in the sections that follow, horizontal, vertical, and rotational motion are decoupled to obtain closed-form solutions. Components are combined:

$$\mathbf{c}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}. \tag{5.5}$$

All derivations assume flat ground with axes x/y and z being parallel and perpendicular to the ground plane respectively. Equations are represented in world coordinates and converted to polynomial form using fifth-order Taylor approximation about $\mathbf{s}_0$.

Our approximate SLIP model results from decoupling and linearizing the inverted pendulum model of Kajita et al. [2001]. In the vertical direction, we also add a spring with constant stiffness. This produces a model with a straightforward interpretation: horizontal motion is governed by inverted pendulum dynamics, while vertical motion is described by spring-mass dynamics.

Horizontal Motion. We first derive equations-of-motion along the horizontal x-axis; since we assume planar dynamics, equations-of-motion in the y direction are identical. Assuming an inverted pendulum of mass m, a height h above the ground, a COP location p_x, and an angle θ from the vertical axis, gravitational forces act in the horizontal direction as $mg\tan\theta$. Using

$\tan\theta = (x - p_x)/h$, we have:

$$m\ddot{x} = mg(x - p_x)/h. \tag{5.6}$$

Because COP motion is linear (Equation 5.3), the solution is given by:

$$x(t) = \beta_1 e^{\alpha t} + \beta_2 e^{-\alpha t} + p_x(t) \tag{5.7}$$

where:

$$\beta_1 = \frac{x_0 - p_{0,x}}{2} + \frac{\dot{x}_0 T - (p_{T,x} - p_{0,x})}{2\alpha T} \tag{5.8}$$

$$\beta_2 = \frac{x_0 - p_{0,x}}{2} - \frac{\dot{x}_0 T - (p_{T,x} - p_{0,x})}{2\alpha T} \tag{5.9}$$

$$\alpha = \sqrt{g/h} \tag{5.10}$$

and $x_0/\dot{x}_0$ are the horizontal COM position/velocity at the beginning of the stance phase, and h is the COM height at the beginning of the stance phase.

Vertical motion. Vertical motion is modeled using a spring mass system, subject to gravitational and spring forces:

$$m\ddot{z} = k(r - z) - mg. \tag{5.11}$$

Because the rest length varies linearly (Equation 5.4), the motion is given by:

$$z(t) = d_1 \cos(\omega t) + d_2 \sin(\omega t) + r(t) - g/\omega^2 \tag{5.12}$$

where

$$d_1 = z_0 - r_0 + \frac{g}{\omega^2} \tag{5.13}$$

$$d_2 = \frac{\dot{z}_0}{\omega} - \frac{r_T - r_0}{T\omega} \tag{5.14}$$

$$\omega = \sqrt{k/m} \tag{5.15}$$

and $z_0/\dot{z}_0$ are the COM position/velocity at the beginning of the stance motion in the vertical direction. In all our examples we use $g = 9.81\ m/s^2$ and $k = 1000\ N/m$.

Heading. To compute the character's heading $\alpha(t)$, we assume the pelvis orientation is independent of linear COM motion. Because the controlled angular acceleration $\ddot{\alpha}$ is constant, we have:

$$\alpha(t) = \alpha_0 + \dot{\alpha}_0 t + \frac{1}{2}\ddot{\alpha}t^2. \tag{5.16}$$

5.2.3 Flight Dynamics and Simulation

The complete state vector during flight is:

$$\mathbf{s}(t) = \begin{bmatrix} \mathbf{c}(t) \\ \dot{\mathbf{c}}(t) \\ \alpha(t) \\ \dot{\alpha}(t) \\ \emptyset \\ \emptyset \end{bmatrix}. \tag{5.17}$$

When the character is in flight, projectile motion equations are used to describe COM motion. Since joint forces have no effect on the COM trajectory or heading, the only control parameter for the flight phase is the duration T:

$$\mathbf{u}_f = \begin{bmatrix} T \end{bmatrix} \tag{5.18}$$

The equations of motion are then:

$$\mathbf{c}(t) = \mathbf{c}_0 + \dot{\mathbf{c}}_0 t - \frac{1}{2}\ddot{\mathbf{c}}t^2 \tag{5.19}$$

$$\alpha(t) = \alpha_0 + \dot{\alpha}_0 t \tag{5.20}$$

and

$$\ddot{\mathbf{c}} = \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} \tag{5.21}$$

is the acceleration due to gravity.

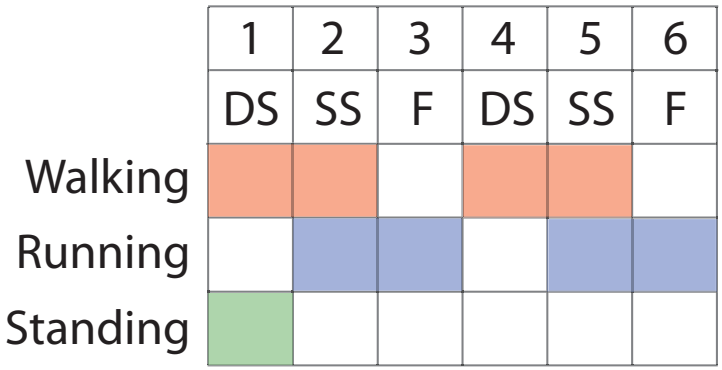

Figure 5.4: Preview optimization schedule. Using a single schedule we capture a large family of locomotion behaviors. Depending on the optimization results different motions are produced (by excluding different phases): Walking alternates between double-stance (DS) and single-stance (SS). Running alternates between SS and Flight (F). Standing only uses a single DS phase. Because we force exchange of support (EOS) between subsequent stance phases, motions such as hopping on one leg require a different schedule (Chapter 7).

5.2.4 Multi-step simulation

Simulation over multiple steps requires concatenating multiple stance and flight phases. Walking emerges as a sequence of alternating double and single-stance phases, whereas running alternates between single-stance and flight (Figure 5.4). In all cases, single-stance phases alternate between left-foot and right-foot stance. We capture multiple types of locomotion with a single, fixed preview schedule of 6 phases (Figure 5.4): double-stance, single-stance, flight, double-stance, single-stance, and flight. Optimizing a preview schedule entails optimizing six sets of control parameters:

$$\mathbf{U} = \left[\begin{array}{cccccc} \mathbf{u}_s^1 & \mathbf{u}_s^2 & \mathbf{u}_f^3 & \mathbf{u}_s^4 & \mathbf{u}_s^5 & \mathbf{u}_f^6 \end{array} \right]. \tag{5.22}$$

In every schedule, only a subset of the phases will occur. Phases assigned a zero duration by the optimization ($T_i = 0$), will be omitted from the final plan. The initial phase must match the initial state of the fully-body character (e.g., if the current character state is single-stance, then the first double-stance phase is skipped). Because it is not possible to transition from flight directly to double-stance (i.e., one foot will always land slightly before the other), at least one of phases 3 and 4 will always be skipped.

When the character is not standing in place, exchange-of-support between left and right stance feet will occur after phases 2 or 3. This requires a new footplant to be selected. We use an

additional 2D control parameter:

$$\mathbf{u}_{foot} = \mathbf{y}_{swing} \tag{5.23}$$

to specify where the swing foot should be planted. Given $\mathbf{y}_{swing}$ and the character's heading α at the beginning of the corresponding phase, the shape of the character's foot is projected onto the environment to determine the corresponding Ω for rest of the schedule.

Given initial conditions, $\mathbf{s}_0$ and control parameters, $\mathbf{U}$, a preview simulation is then performed by applying equations-of-motion for each phase sequentially. Concatenating the individual trajectories gives the low-dimensional motion over the entire preview schedule:

$$\mathbf{S}(t) = \left[\begin{array}{cccccc} \mathbf{s}_s^1 & \mathbf{s}_s^2 & \mathbf{s}_f^3 & \mathbf{s}_s^4 & \mathbf{s}_s^5 & \mathbf{s}_f^6 \end{array} \right]. \tag{5.24}$$

Since each phase is polynomial, the complete sequence is piecewise polynomial. The total duration of the schedule is the sum of the individual durations, $T_{sched} = \sum_i T_i$.

To avoid impulsive changes in motion, we enforce continuity in control parameters across consecutive stance phases. Specifically, $\mathbf{p}_T$ for phase i must equal $\mathbf{p}_0$ for phase $i+1$; likewise r_T for phase i must equal r_0 for phase $i+1$. As a result, the preview control parameters $\mathbf{U}$ consists of 23 free parameters.

5.3 Preview Optimization

Given initial conditions $\mathbf{s}_0$, the preview optimization selects optimal control parameters, $\mathbf{U}^\star$ for the preview schedule, relative to a set of user-specified energy terms $g_i(\mathbf{S})$:

$$\mathbf{U}^\star = \arg\min_{\mathbf{U}} \sum_i w_i g_i(\mathbf{S}) \tag{5.25}$$

Because the landscape of our objective has many local minima, we solve the optimization using Covariance Matrix Adaptation (CMA) [Hansen 2006]. To speed up the optimization, we reuse the solution from the previous timestep as the initial guess for the current solution. In our implementation, we run CMA with a maximum of 5000 objective function evaluations and a population size of 15. The projectile avoidance examples (Section 5.5) uses up to 10000 evaluations.

As discussed below, two types of terms are included in the preview optimization: goal objectives and modeling objectives. The optimal control parameters $\mathbf{U}^\star$ yield an optimal simulation $\mathbf{S}^\star$.

5.3.1 Goal Objectives

Using goal objectives, users can specify desired properties of motion. Our system uses four types of goal objectives: i) step duration, ii) step length, iii) pelvis heading, and iv) COM height. Throughout this section, we denote the first and second foot step of the schedule using A and B respectively. See Table 5.1 for typical settings used in our examples.

We seek solutions with steps of user-specified duration T_{step}. Deviations are penalized using:

$$g_{steptime} = (T_A - T_{step})^2 + (T_B - T_{step})^2 \tag{5.26}$$

where the total duration of each step, T_A and T_B, are the sums of their respective double and single-stance durations.

To encourage steps of a particular length, d_{step}, and in the desired direction of travel, $\mathbf{d}$, we use:

$$g_{stepdist} = (d_A - d_{step})^2 + (d_B - d_{step})^2 \tag{5.27}$$

where

$$d_A = (\mathbf{c}_A - \mathbf{c}_0)^T \mathbf{d} \tag{5.28}$$

$$d_B = (\mathbf{c}_B - \mathbf{c}_A)^T \mathbf{d} \tag{5.29}$$

are the length of the first and second steps, projected in the desired direction of travel, and $\mathbf{c}_i$ is the Cartesian location of the COM at different points in the preview.

Since heading changes cannot be achieved instantaneously, we determine error in orientation at the end of the preview using:

$$g_{heading} = (\alpha - \alpha_d)^2 \tag{5.30}$$

where α_d is the desired character heading and α is the final pelvis orientation determined according to Equations 5.16 and 5.20. This term allows a user to steer the character.

Transitions between walking and jumping arise based on foothold selection and user constraints. To encourage motions with pronounced ballistic phases, we include an objective:

$$g_{com} = (h_\epsilon - h_d)^2 \tag{5.31}$$

$$h_\epsilon = \max\left(h_{apex} - h_{start}, h_{apex} - h_{end}\right). \tag{5.32}$$

h_{start}, h_{end}, and h_{apex} are the COM heights at the start, end, and apex of the first flight phase, respectively. h_d is expressed relative to the COM standing height.

5.3.2 Modeling Objectives

Modeling objectives play an important role in compensating for simplifications made by the decoupled preview dynamics model. We use three modeling objectives: i) COM acceleration, ii) leg length, and iii) foot position relative to hips. Although modeling objectives use error integrals over the preview window, these quantities can be calculated in closed-form, due to the piecewise-polynomial COM preview motion (Section 5.2.2).

Because we optimize a simplified passive model, with no explicit force inputs, it does not make sense to measure stride power. Instead, we seek smooth motions with small COM accelerations:

$$g_{accel} = \int ||\ddot{\mathbf{c}}(t)||^2 \, dt. \tag{5.33}$$

A key simplification made by our stance model is that COM height is constant during stance. Though this simplifies the mathematical preview model description, it can produce unnatural "flat walking" motions. Natural walking is characterized by an arc-like motion of the stance leg about the COP. To encourage this type of motion, we penalize motions with large variations from the target leg length:

$$g_{leg} = \sum_{i \in \{R,L\}} \int (L_i^2 - L_r^2)^2 \, dt \tag{5.34}$$

where $L_i = \left|\left| \mathbf{y}^i_{hip} - \mathbf{y}^i_{ankle} \right|\right|$ is the length of leg i, measured between the hip and the ankle (Figure 5.3) and L_r is the character's leg length when standing. The positions $\mathbf{y}_{hip}$ and $\mathbf{y}_{ankle}$ are defined by fixed horizontal offsets from $\mathbf{c}$ and $\mathbf{y}_{swing}$, respectively. This term also helps avoid drift, keeping the leg length "centered" near its nominal value. Weights for this objective term

are kept small, since motions such as running and jumping require substantial leg compression.

In our preview model, we assume pelvis orientation in the axial plane is completely decoupled from forward motion. Even for characters with small hips, changes to pelvis orientation will require leg-length changes. To compensate for this simplification and to generate motions with the feet roughly pointed in the same direction as the pelvis, we use an objective:

$$g_{hip} = \sum_{i \in \{R,L\}} \int ||\mathbf{y}'_{hip} - \mathbf{y}'_{ankle}||^2 \, dt \tag{5.35}$$

where $\mathbf{y}'_{hip}$ and $\mathbf{y}'_{ankle}$ are the 2D projections of the hip and ankle position onto the ground plane. This term encourages the hip to stay over the ankle.

5.3.3 Constraints

To ensure solutions calculated for the simplified model are valid when mapped to the full character, several geometric constraints, based on target character skeletal properties, are enforced. With the exception of leg length, all of the constraints are expressed in CMA as variable bounds constraints. A sigmoidal soft-constraint [Liu et al. 2005] is used for leg length constraints.

The COP locations $\mathbf{p}_0/\mathbf{p}_T$ must lie within the base-of-support during each stance phase. The locations $\mathbf{p}_0/\mathbf{p}_T$ are represented by bilinear coordinates within the contact quadrilateral. During double-stance, an oriented quadrilateral is fit to the convex hull surrounding Ω_L and Ω_R.

We limit the model's telescoping leg length:

$$L_i \in [\, L_{min}, L_{max} \,] \tag{5.36}$$

to ensure leg length limits imposed by the full character are not violated. L_{max} is the maximal hip to ankle distance of the full character when the legs are fully extended and L_{min} is the same distance when the legs are fully contracted.

Each component of the footstep location is constrained to lie near the COM:

$$||\mathbf{c}_{\parallel} - \mathbf{y}_{swing}||_{\infty} < \frac{1}{2} L_r \tag{5.37}$$

where $\mathbf{c}_{\parallel}$ are the COM coordinates projected on the ground plane and L_r is the character's leg

Table 5.1: Controller Parameter Values. Nominal values are based on observed human motion. Typical preview optimization weights are: $w_{steptime} = 10$, $w_{stepdist} = 10$, $w_{heading} = 1$, $w_{com} = 0$, $w_{accel} = 0.05$, $w_{leg} = 10$, $w_{hip} = 10$.

Action	$d_{step}\,[m]$	$T_{step}\,[s]$	w_{com}	$h_d\,[m]$
Walk	0.75	0.6	0	0
Run/Jump	1.05	0.4	> 0	$0 - 0.5$

length when standing. The L_∞ norm is used to express Equation 5.37 as bounds constraints.

Each phase must have a non-negative duration $T \geq 0$.

5.4 Full-Body Controller

Once $\mathbf{S}^\star$ has been selected, a second optimization computes full-body joint torques $\boldsymbol{\tau}$ for the current time instant. The only quantities used from $\mathbf{S}^\star$ are the instantaneous second derivatives $(\ddot{\mathbf{c}}_d, \ddot{\alpha})$ at the start time $t = 0$, and the target footplant $\mathbf{y}_{swing}$.

Given the optimal preview motion $\mathbf{S}^\star$, we calculate:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{5.38}$$

where $\boldsymbol{\tau}$ are the full-body control torques, $\ddot{\mathbf{q}}$ is the generalized accelerations of the body, and $\boldsymbol{\lambda}$ are linearized friction-cone basis weights, used to estimate contact forces. We formulate control synthesis as a weighted optimization (Section 6.2).

At each time instant, the optimal $\mathbf{x}$ is recalculated; the resulting accelerations $\ddot{\mathbf{q}}$ are then integrated. The dynamics constraints, character model, and integration are implemented as in Chapter 4. We do not use constrained prioritized optimization (Section 6.3), although we suspect that it could provide better stylistic control and robustness.

The optimization uses several objectives. The first two objectives aim to have the COM and pelvis move in the direction suggested by the plan. To do this, we use objectives $E_{com} = ||\ddot{\mathbf{c}}_r - \ddot{\mathbf{c}}||^2$ and $E_{heading} = (\ddot{\alpha}_r - \ddot{\alpha})^2$, where $\ddot{\mathbf{c}}_r$ and $\ddot{\alpha}_r$ are computed by analytic differentiation of the current polynomial plan representation. Feet are moved to the target footholds, $\mathbf{y}_{swing}$, using the target

objective E_{swing} described in Chapter 3. Unlike locomotion controllers from Chapter 4, it is not necessary to hand-design gait-specific COM trajectories or foot targets.

Another objective, E_{hip}, implemented as a setpoint objective (Chapter 4), favors motions with similar hip-to-COM distance as the simplified model. Nominal hip-to-COM distance is calculated at the beginning of the simulation, when the character is in its reference standing posture, and assumed to remain constant. This helps us come up with a better measure of the leg length commanded by the SLIP. This quantity is expressed in the heading reference frame, to encourage the legs and feet to point in the desired direction of travel.

Additional objective and stabilization terms are taken from (Chapter 4): angular momentum regulation (E_{AM}), rest pose (E_{rest}), head stabilization (E_{head}), and foot contact during stance ($E_{contact}$). We do not include stylistic terms, such as arm torque minimization, as we found that these can cause problems during sharp turns. Angular momentum (AM) regulation performs two roles in this optimization. First, as in Chapter 4, AM regulation is useful for balance and stability. Additionally, because our simplified model does not represent rotational motion outside the axial plane, it cannot accurately preview motion with significant AM. Hence, keeping AM small for the full-body movement helps prevent deviations between the models.

It is also possible to compute the optimal preview less frequently, since, in the absence of unexpected events, the preview optimization is normally consistent across a large number of frames. This improves performance. In addition to planning at integer multiples of the integration timestep, we find it useful to allow replanning at contact events (e.g., change of support).

5.5 Results

Using the described formulation, we generate a diverse set of robust locomotion behaviors. Generating distinct behaviors, such as walking, running, and standing, as well as transitions between these behaviors, is achieved by specifying only a high-level set of task goals. No state machines, special treatment of transitions, or motion capture data are required to generate these motions. Resulting motions are shown in the accompanying video.

Our system uses the simulator described in Chapter 4, which uses an inelastic ($\epsilon_{rest} = 0$) LCP-based contact model with Coulomb friction ($\mu = 1$). Our human model has 37 degrees of freedom (Figure 5.3). All limbs are assumed to have constant density. Table 5.1 lists typical

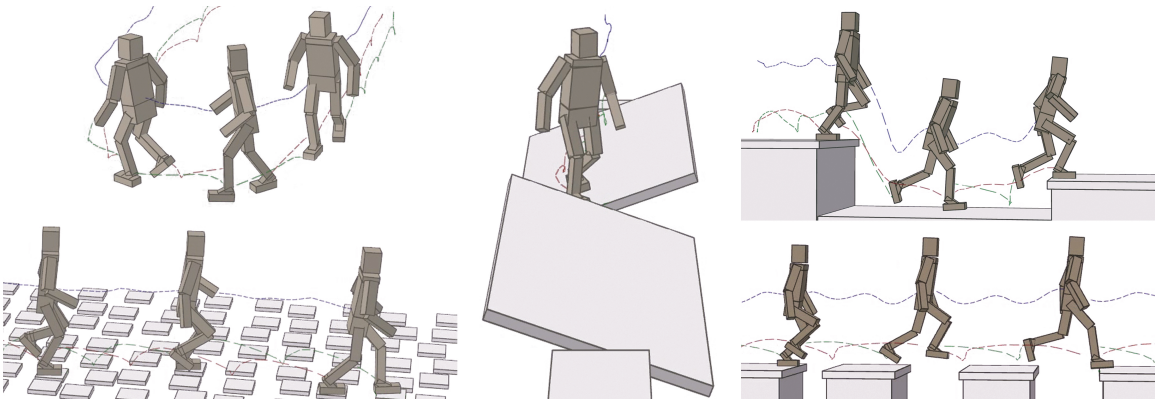

Figure 5.5: Uneven Terrain Capabilities. In addition to walking, running, and turning, low-dimensional planning allows traversal of varied terrain including: drops (1 m), steps (0.5 m), stepping stones, gaps (0.25-0.9 m), and inclines ($\pm 15^o$). See the accompanying video for a complete set of examples.

parameters used in our system to achieve different types of motion.

Performance. When updating dynamics and control at 250Hz, our unoptimized implementation runs at approximately 15% real-time on a Intel i7 920 2.67 GHz processor. Planning less frequently significantly improves performance. To quantify impact of planning frequency on level-ground walking, we test two algorithm variants (Figure 5.6). First, we restrict preview computation to occur every 1, 5, 10, 25, 50, 75, and 100 integration steps. At each step, a COM setpoint objective (Chapter 4) is used with reference acceleration $\ddot{\mathbf{c}}_r$ computed by differentiating the plan, and with gain $k_p^{com} = 1000$. All other aspects of the control remain unchanged. Second, we repeat the first experiment, but also allow replanning at all contact events.

Less-frequent planning significantly improves performance, but decreased robustness. Without allowing replanning at contact events, we were not able to plan less frequently than every 100 steps. When limiting replanning to only occur at contact events, the average speed is 55% real-time; remaining computation is for our simulator and the full-body controller. Replanning only at contacts yields significant speedups with minimal motion changes. In this case, replanning constitutes a negligible portion of computation time.

To further improve performance, our method might also benefit from more sophisticated optimization methods, such as Basin-CMA [Wampler and Popović 2009], which could leverage the analytically-differentiable objective provided by our method. Planning frequency could also be dynamically adjusted based on detected disturbances or preview deviations.

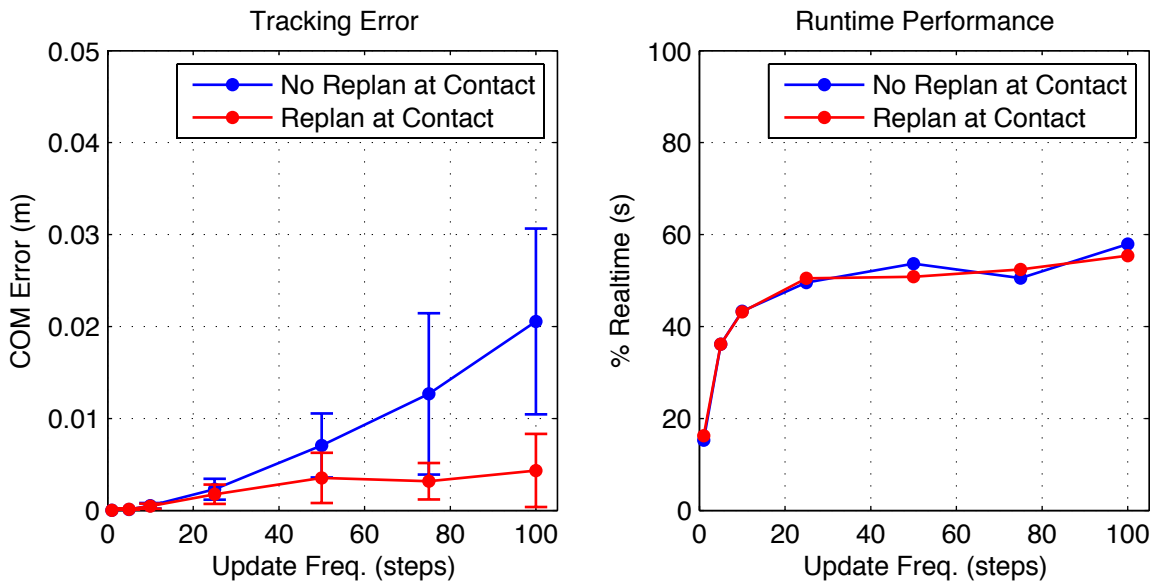

Figure 5.6: Performance vs. Update Frequency. Mean COM tracking error and runtime speed, as a function of replanning frequency, for level-ground walking. Standard deviations are shown for tracking error. COM error is the mean cumulative COM tracking error over each replanning window. Planning was performed at specified multiples of the integration timestep (blue). To improve stability, we also allowed replanning at contact transitions (red). Replanning only at contacts yields speedups with only minimal tracking error.

Standing. To have the character stand in place, we set the desired step-size $d_{step} = 0$ and deactivate the step-duration objective $w_{steptime} = 0$. This results in a preview consisting of a single arbitrarily long double-stance phase. If the desired heading, α_d, is changed during standing, the character will take the necessary steps to avoid errors from g_{hip}, while also satisfying $g_{heading}$. Small changes in desired orientation are handled by the torso, while large turns coordinate leg joints. In addition to smoothing locomotion gaits, using a linear COP model (Equation 5.4) enhances standing by allowing perturbations to be handled using in-place weight shifts, rather than requiring many small compensatory steps.

Walking, Running, and Jumping. Walking and running are performed by adjusting step parameters (Table 5.1). Transitions between these two locomotion modes occurs automatically as parameters are changed. To force motion with marked ballistic periods, the COM height objective (Equation 5.32) is activated and h_d is set to a nonzero value. Alternatively, the weighting of g_{accel} can be decreased. This will allow larger deviations in c̈.

Uneven Terrain Navigation. Because our preview model plans into the future, control developed for flat ground can also be used for uneven terrain without changes. The controller can successfully traverse many types of uneven and constrained terrain, including gaps, inclines, stairs, large drops, and stepping stones (Figure 5.5). A demonstration of the character traversing an obstacle course with many of these elements is shown in Figure 5.1. Terrain is represented as a height field, which is provided to the preview optimization to select good footplants $\mathbf{y}_{swing}$. For large gaps, we have found it useful to provide the optimization with a signed distance field describing favored foot plant locations on the terrain. Specifically, when possible we prefer the character step in the middle of a platform, away from edges. Currently our system handles inclines of $\pm 15^o$. Larger inclines violate flat ground modeling assumptions of our approximate SLIP model.

External Disturbances and Projectile Avoidance. Previous work has quantified controller robustness through external perturbation tests (e.g., throwing projectiles at the character [Abe et al. 2007; Coros et al. 2009]). In the accompanying video we demonstrate controller robustness, by testing external forces ranging from 100-500 N applied for 0.1-0.2 sec. to the character's head, at both random and cyclic intervals. In addition to applying forces when the character is in stance, we also test the challenging case of in-air disturbances. In both cases, the planner generates steps to avoid failure.

Our controller can also be robust to projectiles by dodging them. We add a projectile avoidance constraint to the planner that constrains the plan to keep the COM above a minimum distance D from the projectile's path:

$$\min_t || \mathbf{p}(t) - \mathbf{c}(t) ||^2 > D^2 \tag{5.39}$$

where t indexes the duration of the plan, and $\mathbf{p}(t)$ is the future path of the projectile. This constraint is enforced by a sigmoidal soft constraint, and the minimum is computed over a set of discrete sample times t. As a result, plans are generated that take necessary evasive maneuvers to step away from projectiles. When the heading objective is deactivated, the character can changes whole-body orientation to more quickly avoid projectiles. When heading is active, avoidance strategies generally involve taking larger steps.

5.6 Summary

This chapter has presented a method for simplifying design of feature-based locomotion controllers. Instead of hand-designing state-machines for different behaviors as described in Chapter 4, we formulate the feature selection process as nonlinear trajectory optimization.

At each time instant the nonlinear optimization proposes a set of feature acceleration targets. Features are selected by evaluating a low-dimensional hybrid dynamical model that predicts how actions taken at the current time instant will impact motion in the near future. To evaluate the model, an objective is used that accounts for user-specified goals and modeling simplifications. Once features are selected, we calculate joint torques, for forward simulation, using methods described in previous chapters.

We are not the first to consider planning of biped motions. However, unlike many previous kinematic (e.g., [Kuffner et al. 2003]) and kinodynamic (e.g., [Chestnutt 2007; Coros et al. 2009]) techniques, we do not rely on discrete search for action selection. Instead, our low-dimensional formulation allows optimizations to remain tractable, while also considering the state-dependent nature of actions. Although we initially pursued this approach to improve foot plant selection on flat ground, we demonstrate good generalization capabilities to many different gaits, terrain types, and constraints (e.g., projectile avoidance).

Chapter 6

Prioritized Optimization

The process of optimizing a collection of objective functions is referred to as *multi-objective optimization* (MOO). Methods for MOO have a long history in economic theory, game theory, and pure mathematics [Marler and Arora 2004]; see [Stadler 1988] for a historical perspective. In addition to practical considerations of efficiency, MOO algorithms must overcome two challenges. First, MOO methods must provide a way to combine multiple, possibly conflicting, objectives. Second, for nonlinear objectives functions, MOO algorithms must choose amongst a possibly large set of solutions.

The most common method of combining multiple objectives is weighted combination (Section 6.2). For problems with quadratic objectives and linear constraints, this amounts to solving a quadratic program (QP). In practice, tuning weights for problems with many objectives can be difficult (Section 4.5). An alternative, to weighted combination, is lexicographic or prioritized optimization (PO). This approach solves objectives in a hierarchical order; at each priority level, objectives are solved subject to problem constraints and equality constraints arising from all higher priority objectives. Most PO methods (e.g., [Kanoun et al. 2009]) use a formulation that systematically increases the number of equality constraints considered at each priority level (Section 6.3.4).

In this chapter, we propose two algorithms that use an alternate strategy. By focusing on positive semi-definite quadratic objectives and linear constraints we can enforce prioritization by solving a series of nested underconstrained linear systems (Section 6.1). Each linear system is solved in the null-space basis of higher-priority problems. When unilateral constraints are included (Section 6.3), each priority level requires a QP to be solved; constraints must also be

reparameterized. Therefore, in our algorithms the number of constraints remains constant. Our algorithms also allow multiple objectives to be combined by weighted combination at any priority level. Hence, proposed algorithms provide a single method of handling both weighted and prioritized optimization problems.

By focusing on quadratic objectives and solving underconstrained problems in a least-square sense, we avoid the having to maintain a set of potential minimizers (as described above for nonlinear objectives). If a feasible minimizer exists it is guaranteed to be the global solution.

6.1 Unconstrained Prioritized Optimization

6.1.1 Problem Statement

Given an ordered list of N objectives, $E_1(\mathbf{x}), \ldots, E_N(\mathbf{x})$, over some variable $\mathbf{x} \in \mathbb{R}^D$, prioritized optimization finds the solution minimizing each objective without conflicting with higher-priority objectives:

$$\begin{aligned} h_i &= \min_{\mathbf{x}} E_i(\mathbf{x}) \\ \text{subject to } & E_k(\mathbf{x}) = h_k, \ \forall k < i \end{aligned} \tag{6.1}$$

For example, suppose we are given three objective functions $E_1(\mathbf{x}), E_2(\mathbf{x})$, and $E_3(\mathbf{x})$. There are three steps to define the solution to this problem. Our first goal is to find the minimum of $E_1(\mathbf{x})$. Since objectives are generally underconstrained, $E_1(\mathbf{x})$ will typically have multiple minimizers. Our next goal is to find which among these valid minimizers have minimal $E_2(\mathbf{x})$. Finally, among this new set of minimizers, we want to return the one which minimizes $E_3(\mathbf{x})$. More formally, we can write the problem as the following sequence of optimizations:

$$h_1 = \min_{\mathbf{x}} E_1(\mathbf{x}) \tag{6.2}$$

$$\begin{aligned} h_2 &= \min_{\mathbf{x}} E_2(\mathbf{x}) \\ &\text{subject to } E_1(\mathbf{x}) = h_1 \end{aligned} \tag{6.3}$$

$$\begin{aligned} h_3 &= \min_{\mathbf{x}} E_3(\mathbf{x}) \\ &\text{subject to } E_1(\mathbf{x}) = h_1, E_2(\mathbf{x}) = h_2 \end{aligned} \tag{6.4}$$

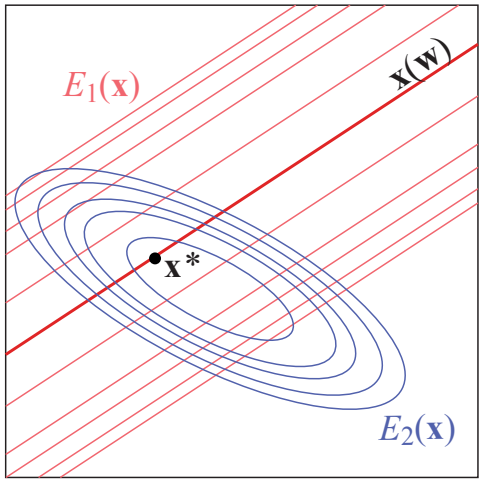

Figure 6.1: Prioritized optimization problem with 2 objectives. $\mathbf{E}_1$ (in red) constrains the minimizer to lie in the family of solutions $\mathbf{x}(\mathbf{w})$ shown as a dark red line. Selecting among valid solutions, the solver identifies the minimizer in the family of solutions $\mathbf{x}(\mathbf{w})$ producing lowest $\mathbf{E}_2$ (shown in blue).

We then return the value $\mathbf{x}^*$ that solves this final optimization. Figure 6.1 illustrates a case with two tasks; the general case of N objectives can be defined as in Equation 6.1.

In principle, prioritized optimization could be solved by N applications of a general constrained optimization algorithm. In this chapter, we focus on the case where all objectives are positive semidefinite quadratics, and thus can be written as:

$$E_i(\mathbf{x}) = || \mathbf{A}_i\mathbf{x} - \mathbf{b}_i ||^2 \tag{6.5}$$

for some matrix $\mathbf{A}_i$ and vector $\mathbf{b}_i$. Given N objectives, our goal is to solve for unknowns $\mathbf{x}$ such that we minimize $E_i(\mathbf{x})$ subject to each of the previous objectives.

In this formulation, each objective can also be thought of as a constraint of the form $E_i(\mathbf{x}) = 0$. Solving the prioritized optimization problem ensures that this constraint will be satisfied when possible, since, in general, E_i is minimized at zero. When a constraint cannot be satisfied (typically because it conflicts with a higher-level constraint), minimizing E_i will minimize the failure of the constraint. Hence, each E_i can be viewed as either a constraint or as an objective.

6.1.2 Solution for Three Objectives

To see how to solve this type of optimization problem, we consider the case of three quadratic objectives $E_1(\mathbf{x}), E_2(\mathbf{x})$, and $E_3(\mathbf{x})$. Solving for $\mathbf{x}$ in a naïve manner would involve solving

a sequence of quadratically-constrained quadratic programs. However, the space of optimal solutions to a positive semidefinite quadratic objective must be a linear subspace.

Specifically, the space of optimal solutions to $E_1 = || \mathbf{A}_1\mathbf{x} - \mathbf{b}_1 ||^2$ can be parameterized by a vector $\mathbf{w}_1$ as:

$$\mathbf{x}(\mathbf{w}_1) = \mathbf{C}_1\mathbf{w}_1 + \mathbf{d}_1 \tag{6.6}$$

where $\mathbf{C}_1 = \text{null}(\mathbf{A}_1)$ is a nullspace basis for $\mathbf{A}_1$, and $\mathbf{d}_1 = \mathbf{A}_1^\dagger\mathbf{b}_1$ is a minimizer of E_1. Any choice of $\mathbf{w}_1$ produces a minimizer of $E_1(\mathbf{x})$. The quantities $\mathbf{C}_1$ and $\mathbf{d}_1$ can be computed using a single Singular Value Decomposition (SVD). Note that their dimensionalities depend on the rank of $\mathbf{A}_1$.

Now we can reparameterize the second objective:

$$E_2(\mathbf{w}_1) = || \mathbf{A}_2\mathbf{x}(\mathbf{w}_1) - \mathbf{b}_2 ||^2 \tag{6.7}$$
$$= || \mathbf{A}_2\mathbf{C}_1\mathbf{w}_1 - (\mathbf{b}_2 - \mathbf{A}_2\mathbf{d}_1) ||^2 \tag{6.8}$$
$$= \left|\left| \bar{\mathbf{A}}_2\mathbf{w}_1 - \bar{\mathbf{b}}_2 \right|\right|^2 \tag{6.9}$$

where:

$$\bar{\mathbf{A}}_2 = \mathbf{A}_2\mathbf{C}_1 \tag{6.10}$$
$$\bar{\mathbf{b}}_2 = \mathbf{b}_2 - \mathbf{A}_2\mathbf{d}_1. \tag{6.11}$$

The solutions to this problem can be described by a smaller subspace, parameterized by a vector $\mathbf{w}_2$ as $\mathbf{w}_1(\mathbf{w}_2) = \mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2$, where $\mathbf{C}_2 = \text{null}(\bar{\mathbf{A}}_2)$ and $\mathbf{d}_2 = \bar{\mathbf{A}}_2^\dagger\bar{\mathbf{b}}_2$. If we only have two objectives, such as the example shown in Figure 6.1, the solution to the optimization problem is $\mathbf{x}^* = \mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1$.

To account for the third objective, we reparametrize E_3 in terms of $\mathbf{w}_2$, using:

$$\mathbf{x}(\mathbf{w}_2) = \mathbf{x}(\mathbf{w}_1(\mathbf{w}_2)) \tag{6.12}$$
$$= \mathbf{C}_1(\mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2) + \mathbf{d}_1 \tag{6.13}$$
$$E_3(\mathbf{w}_2) = || \mathbf{A}_3\mathbf{x}(\mathbf{w}_2) - \mathbf{b}_3 ||^2 \tag{6.14}$$
$$= \left|\left| \bar{\mathbf{A}}_3\mathbf{w}_2 - \bar{\mathbf{b}}_3 \right|\right|^2 \tag{6.15}$$

where

$$\bar{\mathbf{A}}_3 = \mathbf{A}_3\mathbf{C}_1\mathbf{C}_2 \tag{6.16}$$

$$\bar{\mathbf{b}}_3 = \mathbf{b}_3 - \mathbf{A}_3(\mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1). \tag{6.17}$$

The subspace of optima for this problem is given by $\mathbf{w}_2(\mathbf{w}_3) = \mathbf{C}_3\mathbf{w}_3 + \mathbf{d}_3$, where $\mathbf{C}_3$ and $\mathbf{d}_3$ are defined as above. Then, any value of $\mathbf{w}_3$ gives an optimal solution for the final objective. Substituting in the original space, we have:

$$\mathbf{x}(\mathbf{w}_3) = \mathbf{x}(\mathbf{w}_1(\mathbf{w}_2(\mathbf{w}_3))) \tag{6.18}$$

$$= \bar{\mathbf{C}}\mathbf{w}_3 + \bar{\mathbf{d}} \tag{6.19}$$

where:

$$\bar{\mathbf{C}} = \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3 \tag{6.20}$$

$$\bar{\mathbf{d}} = \mathbf{C}_1\mathbf{C}_2\mathbf{d}_3 + \mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1. \tag{6.21}$$

Hence, a solution to the entire prioritized optimization problem is given by $\mathbf{x}^* = \bar{\mathbf{d}}$.

6.1.3 General Solution

We can generalize to the case of N objectives as follows. First, we define an initial subspace equivalent to the full space $\mathbf{x}(\mathbf{w}_0) = \mathbf{w}_0$, so that $\mathbf{C}_0 = \mathbf{1}_{D\times D}$ and $\mathbf{d}_0 = \mathbf{0}_D$. Then for all $1 \leq i \leq N$, we have:

$$\bar{\mathbf{A}}_i = \mathbf{A}_i \prod_{k=0}^{i-1} \mathbf{C}_k \tag{6.22}$$

$$\bar{\mathbf{b}}_i = \mathbf{b}_i - \mathbf{A}_i \sum_{j=1}^{i-1} \left(\prod_{k=0}^{j-1} \mathbf{C}_k \right) \mathbf{d}_j \tag{6.23}$$

$$\mathbf{C}_i = \text{null}(\bar{\mathbf{A}}_i) \tag{6.24}$$

$$\mathbf{d}_i = \bar{\mathbf{A}}_i^{\dagger}\bar{\mathbf{b}}_i \tag{6.25}$$

At each step, the substitution of the subspace is of the form $E_i(\mathbf{w}_i) = \left|\left| \bar{\mathbf{A}}_i\mathbf{w}_i + \bar{\mathbf{b}}_i \right|\right|^2$. The solution subspace for this problem is $\mathbf{w}_i(\mathbf{w}_{i-1}) = \mathbf{C}_i\mathbf{w}_{i-1} + \mathbf{d}_i$. The substitution of $\mathbf{d}_i$ back into the original space is:

$$\bar{\mathbf{d}}_i = \sum_{j=1}^{i} \left(\prod_{k=0}^{j-1} \mathbf{C}_k \right) \mathbf{d}_j \tag{6.26}$$

yielding the final solution to the entire problem $\mathbf{x}^* = \bar{\mathbf{d}}_N$.

6.1.4 Recursive Algorithm

Equations 6.22 and 6.25 provide a recursive description of the unconstrained prioritized solver for quadratic constraints. However, they contain a great deal of redundant computation. For example, the product:

$$\bar{\mathbf{C}}_i = \prod_{j=0}^{i} \mathbf{C}_j \tag{6.27}$$

is computed multiple times. To avoid this duplication, we substitute (6.27) into (6.26), yielding:

$$\bar{\mathbf{d}}_i = \sum_{j=1}^{i} \bar{\mathbf{C}}_{j-1}\mathbf{d}_j \tag{6.28}$$

$$= \bar{\mathbf{d}}_{i-1} + \bar{\mathbf{C}}_{i-1}\mathbf{d}_i \tag{6.29}$$

$$= \bar{\mathbf{d}}_{i-1} + \bar{\mathbf{C}}_{i-1}\bar{\mathbf{A}}_i^{\dagger}\bar{\mathbf{b}}_i \tag{6.30}$$

Using Equations 6.27 and 6.28 we can rewrite $\bar{\mathbf{A}}_i$ and $\bar{\mathbf{b}}_i$ as:

$$\bar{\mathbf{A}}_i = \mathbf{A}_i\bar{\mathbf{C}}_{i-1} \tag{6.31}$$

$$\bar{\mathbf{b}}_i = \mathbf{b}_i - \mathbf{A}_i\bar{\mathbf{d}}_{i-1} \tag{6.32}$$

Using the above and defining accumulator matrices $\bar{\mathbf{C}}$ and $\bar{\mathbf{d}}$ yields the recursive algorithm:

Algorithm 1: Unconstrained Quadratic Prioritized Optimization Solver

1 $\bar{\mathbf{C}} \leftarrow \mathbf{I}$, $\bar{\mathbf{d}} \leftarrow 0$
2 **for** $i = 1$ *to* N **do**
3 $\quad \bar{\mathbf{A}}_i \leftarrow \mathbf{A}_i \bar{\mathbf{C}}$
4 $\quad \bar{\mathbf{b}}_i \leftarrow \mathbf{b}_i - \mathbf{A}_i \bar{\mathbf{d}}$
5 $\quad \bar{\mathbf{d}} \leftarrow \bar{\mathbf{d}} + \bar{\mathbf{C}} \bar{\mathbf{A}}_i^{\dagger} \bar{\mathbf{b}}_i$
6 $\quad$ **if** $\bar{\mathbf{A}}_i$ *is full rank* **then**
7 $\quad\quad$ return $\bar{\mathbf{d}}$
8 $\quad$ **end**
9 $\quad \bar{\mathbf{C}} \leftarrow \bar{\mathbf{C}} \, \mathrm{null}(\bar{\mathbf{A}}_i)$
10 **end**
11 return $\bar{\mathbf{d}}$

If the objective functions are designed well, line 11 will never be reached. For example, if $\mathbf{A}_N$ is full-rank, then $\bar{\mathbf{A}}_N$ must also be full-rank.

The nullspace and pseudoinverse computations using SVD require user-specified tolerances. These tolerances impact solution smoothness. We use:

$$\mathrm{tol}_{pinv} = \sqrt{\epsilon} \tag{6.33}$$

$$\mathrm{tol}_{null} = \max(m, n) \max(\mathbf{s}) \epsilon \tag{6.34}$$

where m and n are the dimensions of each $\bar{\mathbf{A}}_i$, $\mathbf{s}$ are its eigenvalues, and ϵ is the machine tolerance for the floating point representation used by the implementation.

The exact complexity of the iterative solver described in Algorithm 1 depends on the number of objectives and rank of each objective. However, since a SVD is required for each task, a worst-case estimate is $O(D^3)$. This is equivalent to other existing task-space control approaches, which typically require a matrix inverse, or pseudoinverse, to compute projection operators [Liegeois 1977; Khatib 1995].

6.2 Weighted Combination

The algorithm described in Section 6.1 combines objectives in a strictly prioritized manner. Alternatively, users may wish to combine multiple objectives by weighted combination:

$$E(\mathbf{x}) = \sum_j^J \alpha_j E_j(\mathbf{x}) \tag{6.35}$$

with weights α_j. Algorithm 1 also supports weighted combination. At any priority level i, multiple objectives $(\mathbf{A}_j, \mathbf{b}_j)$ can be combined into a single quadratic as:

$$\mathbf{A}_i = \begin{bmatrix} \beta_1 \mathbf{A}_1 \\ \vdots \\ \beta_J \mathbf{A}_J \end{bmatrix} \qquad \mathbf{b}_i = \begin{bmatrix} \beta_1 \mathbf{b}_1 \\ \vdots \\ \beta_J \mathbf{b}_J \end{bmatrix} \tag{6.36}$$

where $\beta_j = \sqrt{\alpha_j}$. All other algorithm details remain unchanged.

Proof.

$$\begin{aligned}
E(\mathbf{x}) &= || \mathbf{A}_i\mathbf{x} - \mathbf{b}_i ||^2 \\
&= \left|\left| \begin{bmatrix} \beta_1 \mathbf{A}_1 \\ \vdots \\ \beta_J \mathbf{A}_J \end{bmatrix} \mathbf{x} - \begin{bmatrix} \beta_1 \mathbf{b}_1 \\ \vdots \\ \beta_J \mathbf{b}_J \end{bmatrix} \right|\right|^2 \\
&= \begin{bmatrix} \beta_1 \mathbf{A}_1\mathbf{x} - \beta_1 \mathbf{b}_1 \\ \vdots \\ \beta_J \mathbf{A}_J\mathbf{x} - \beta_J \mathbf{b}_J \end{bmatrix}^T \begin{bmatrix} \beta_1 \mathbf{A}_1\mathbf{x} - \beta_1 \mathbf{b}_1 \\ \vdots \\ \beta_J \mathbf{A}_J\mathbf{x} - \beta_J \mathbf{b}_J \end{bmatrix} \\
&= \sum_j^J (\beta_j \mathbf{A}_j\mathbf{x} - \beta_j \mathbf{b}_j)^T (\beta_j \mathbf{A}_j\mathbf{x} - \beta_j \mathbf{b}_j) \\
&= \sum_j^J \beta_j^2 (\mathbf{A}_j\mathbf{x} - \mathbf{b}_j)^T (\mathbf{A}_j\mathbf{x} - \mathbf{b}_j) \\
&= \sum_j^J \alpha_j || \mathbf{A}_j\mathbf{x} - \mathbf{b}_j ||^2 = \sum_j^J \alpha_j E_j(\mathbf{x})
\end{aligned}$$

□

6.3 Constrained Prioritized Optimization

6.3.1 Problem Statement

In Section 6.1 we present a recursive algorithm for unconstrained prioritized optimization. The algorithm supports both strict prioritization and weighted combination. Although the algorithm was designed for unconstrained problems, prioritization can be leveraged to enforce equality constraints:

$$C(\mathbf{x}) = \mathbf{0} \tag{6.37}$$

by specifying appropriate objectives at the highest level of priority.

In practice, it may also be desirable to enforce inequality constraints:

$$\mathbf{D}\mathbf{x} + \mathbf{f} \geq \mathbf{0} \tag{6.38}$$

arising from bounds on $\mathbf{x}$. In this section we extend the algorithm from Section 6.1 to such cases.

Given an ordered list of N objectives, $E_1(\mathbf{x}), \ldots, E_N(\mathbf{x})$, constrained prioritized optimization finds the solution minimizing each objective without conflicting with higher-priority objectives and top-level constraints. This is defined recursively as:

$$\begin{aligned} h_i = & \min_{\mathbf{x}} E_i(\mathbf{x}) \\ \text{subject to } & E_k(\mathbf{x}) = h_k, \ \forall k < i \\ & C(\mathbf{x}) = \mathbf{0} \\ & \mathbf{D}\mathbf{x} + \mathbf{f} \geq \mathbf{0}. \end{aligned} \tag{6.39}$$

6.3.2 Solution for Three Objectives

Suppose we have three prioritized objectives, $E_1(\mathbf{x})$, $E_2(\mathbf{x})$, and $E_3(\mathbf{x})$. The space of optimal solutions to $E_1(\mathbf{x}) = || \mathbf{A}_1\mathbf{x} - \mathbf{b}_1 ||^2$, subject to the unilateral constraints, can be parameterized

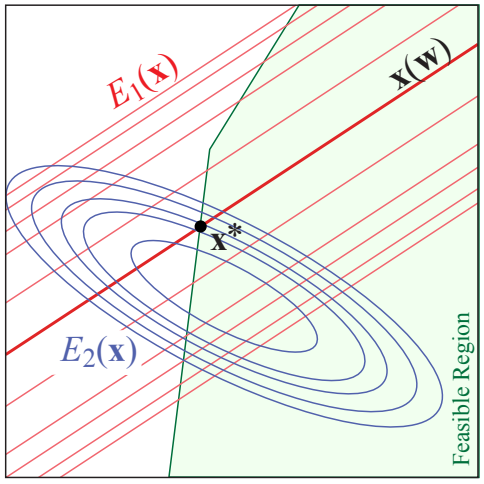

Figure 6.2: Constrained prioritized optimization problem with two objectives. The minima of $E_1(\mathbf{x})$ (in red) lie in a linear subspace $\mathbf{x}(\mathbf{w})$, shown as a dark red line. The solution must lie on this subspace. The dynamics constraints require the solution to lie the green polytope. The solution that minimizes $E_2(\mathbf{x})$, subject to these two constraints, is given by $\mathbf{x}^*$.

by a vector $\mathbf{w}_1$ as:

$$\mathbf{x}(\mathbf{w}_1) = \mathbf{C}_1\mathbf{w}_1 + \mathbf{d}_1 \tag{6.40}$$

where $\mathbf{C}_1 = \text{null}(\mathbf{A}_1)$ is a nullspace basis for $\mathbf{A}_1$, and $\mathbf{d}_1$ is a feasible[1] minimizer of $E_1(\mathbf{x})$. Then, any feasible choice of $\mathbf{w}_1$ produces a minimizer of $E_1(\mathbf{x})$, although not all values of $\mathbf{w}_1$ yield feasible solutions. The solution $\mathbf{d}_1$ can be obtained by solving a QP, and $\mathbf{C}_1$ can be computed using Singular Value Decomposition (SVD). Figure 6.2 illustrates the solution for the case of two objectives.

This reparameterization may then be repeated recursively. Substituting $\mathbf{x}(\mathbf{w}_1)$ into the second objective gives:

$$\begin{aligned} E_2(\mathbf{w}_1) &= || \mathbf{A}_2\mathbf{x}(\mathbf{w}_1) - \mathbf{b}_2 ||^2 \\ &= \left|\left| \bar{\mathbf{A}}_2\mathbf{w}_1 - \bar{\mathbf{b}}_2 \right|\right|^2 \qquad (6.41) \\ \bar{\mathbf{A}}_2 &= \mathbf{A}_2\mathbf{C}_1 \qquad (6.42) \\ \bar{\mathbf{b}}_2 &= \mathbf{b}_2 - \mathbf{A}_2\mathbf{d}_1. \qquad (6.43) \end{aligned}$$

The second objective is now reparameterized so as to restrict it to the space of solutions to the

[1] We use "feasible" for any point that satisfies the unilateral constraints.

first objective. Likewise, substituting $\mathbf{x}(\mathbf{w}_1)$ into the unilateral constraints gives reduced constraints:

$$\mathbf{c}(\mathbf{w}_1) = \mathbf{D}\mathbf{C}_1\mathbf{w}_1 + \mathbf{D}\mathbf{d}_1 + \mathbf{f} \geq 0 \tag{6.44}$$

Now, the solutions to this problem can be described by a smaller subspace, parameterized by a vector $\mathbf{w}_2$ as $\mathbf{w}_1(\mathbf{w}_2) = \mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2$, where $\mathbf{C}_2 = \text{null}(\bar{\mathbf{A}}_2)$ and

$$\mathbf{d}_2 = \arg\min E_2(\mathbf{w}_1) \tag{6.45}$$

$$\text{subject to } \mathbf{c}(\mathbf{w}_1) \geq 0\ . \tag{6.46}$$

We finally reparametrize the third problem in terms of $\mathbf{w}_2$, using:

$$\mathbf{x}(\mathbf{w}_2) = \mathbf{x}(\mathbf{w}_1(\mathbf{w}_2)) = \mathbf{C}_1(\mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2) + \mathbf{d}_1 \tag{6.47}$$

$$E_3(\mathbf{w}_2) = \left|\left|\ \bar{\mathbf{A}}_3\mathbf{w}_2 - \bar{\mathbf{b}}_3\ \right|\right|^2 \tag{6.48}$$

$$\mathbf{c}(\mathbf{w}_2) = \mathbf{D}(\mathbf{C}_1(\mathbf{C}_2\mathbf{w}_2 + \mathbf{d}_2) + \mathbf{d}_1) + \mathbf{f} \tag{6.49}$$

where:

$$\bar{\mathbf{A}}_3 = \mathbf{A}_3\mathbf{C}_1\mathbf{C}_2 \tag{6.50}$$

$$\bar{\mathbf{b}}_3 = \mathbf{b}_3 - \mathbf{A}_3(\mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1). \tag{6.51}$$

The subspace of optima for this problem is given by $\mathbf{w}_2(\mathbf{w}_3) = \mathbf{C}_3\mathbf{w}_3 + \mathbf{d}_3$, where $\mathbf{C}_3$ and $\mathbf{d}_3$ are defined as above. Then, any value of $\mathbf{w}_3$ gives an optimal solution for the final objective. Substituting in the original space, we have:

$$\mathbf{x}(\mathbf{w}_3) = \mathbf{x}(\mathbf{w}_1(\mathbf{w}_2(\mathbf{w}_3))) \tag{6.52}$$

$$= \bar{\mathbf{C}}\mathbf{w}_3 + \bar{\mathbf{d}} \tag{6.53}$$

where:

$$\bar{\mathbf{C}} = \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3 \tag{6.54}$$

$$\bar{\mathbf{d}_3} = \mathbf{C}_1\mathbf{C}_2\mathbf{d}_3 + \mathbf{C}_1\mathbf{d}_2 + \mathbf{d}_1. \tag{6.55}$$

Hence, a solution to the entire constrained prioritized optimization problem is given by $\mathbf{x}_3^* = \bar{\mathbf{d}}_3$.

Algorithm 2: Constrained Quadratic Prioritized Solver

1 $\bar{\mathbf{C}} \leftarrow \mathbf{I}$, $\bar{\mathbf{d}} \leftarrow 0$
2 **for** $i = 1$ *to* N **do**
3 $\quad \bar{\mathbf{A}}_i \leftarrow \mathbf{A}_i \bar{\mathbf{C}}$
4 $\quad \bar{\mathbf{b}}_i \leftarrow \mathbf{b}_i - \mathbf{A}_i \bar{\mathbf{d}}$
5 $\quad \mathbf{d}_i \leftarrow \arg\min_{\mathbf{w}} \left|\left| \bar{\mathbf{A}}_i \mathbf{w} - \bar{\mathbf{b}}_i \right|\right|^2$
6 $\quad\quad$ subject to $\mathbf{D}_0 \bar{\mathbf{C}} \mathbf{w} + \mathbf{D}_0 \bar{\mathbf{d}} + \mathbf{f}_0 \geq 0$
7 $\quad$ **if** *problem is infeasible* **then**
8 $\quad\quad$ return $\bar{\mathbf{d}}$
9 $\quad$ **end**
10 $\quad \bar{\mathbf{d}} \leftarrow \bar{\mathbf{d}} + \bar{\mathbf{C}} \mathbf{d}_i$
11 $\quad$ **if** $\bar{\mathbf{A}}_i$ *is full rank* **then**
12 $\quad\quad$ return $\bar{\mathbf{d}}$
13 $\quad$ **end**
14 $\quad \bar{\mathbf{C}} \leftarrow \bar{\mathbf{C}}\, \mathrm{null}(\bar{\mathbf{A}}_i)$
15 **end**
16 return $\bar{\mathbf{d}}$

6.3.3 Recursive Algorithm

An efficient algorithm for computing the solution for N objectives is given in Algorithm 2. This is a straightforward generalization of the procedure described in Section 6.1.4, the only difference is that inequality constraints are now enforced when solving for $\mathbf{d}_i$ in lines 5-6. If the prioritized optimization problem is well-posed, then line 16 will never be reached. For example, if $\mathbf{A}_N$ is full-rank, then $\bar{\mathbf{A}}_N$ must also be full-rank.

We have tested this algorithm with several QP solvers, including MOSEK's [MOSEK] interior-point method and the active-set solver in QPC [Wills]. We find that MOSEK works for all problems, but is somewhat slower that QPC. QPC is faster, but may not strictly satisfy all inequality constraints.

6.3.4 Alternate Strategy

Section 6.3.3 presents a strategy for solving Equation 6.39, based on reparametrizing objectives at each priority level. Kanoun et al. [2009] propose an alternate strategy based on satisfying an increasing number of equality constraints. Their method is summarized below.

Let $\mathbf{x}_k^*$ be an optimizer of the k-th recursive optimization problem, and $h_k = E_k(\mathbf{x}_k^*)$. Then the

subsequent constraint $E_k(\mathbf{x}) = h_k$ is equivalent to $\mathbf{A}_k\mathbf{x} - \mathbf{b}_k = \mathbf{A}\mathbf{x}_k^* - \mathbf{b}_k$ or, simply:

$$\mathbf{A}_k \left(\mathbf{x} - \mathbf{x}_k^*\right) = \mathbf{0} \tag{6.56}$$

Proof.

Claim 1. $\mathbf{x}$ *is a miminizer of* $E_k(\mathbf{x}) = \| \mathbf{A}_k\mathbf{x} - \mathbf{b}_k \|^2$ *if and only if* $\mathbf{A}_k \left(\mathbf{x} - \mathbf{x}_k^\star\right) = \mathbf{0}$

Part 1. The complete family of solutions is:

$$\mathbf{x}(\mathbf{w}) = \mathbf{x}_k^\star + \mathbf{C}_k\mathbf{w} \tag{6.57}$$

where $\mathbf{C}_k$ is the nullspace of $\mathbf{A}_k$ (i.e., $\mathbf{A}_k\mathbf{C}_k = \mathbf{0}$). Substituting Equation 6.57 into Equation 6.56:

$$\begin{aligned} \mathbf{A}_k \left(\mathbf{x}_k^\star + \mathbf{C}_k\mathbf{w} - \mathbf{x}_k^\star\right) &= \mathbf{0} \\ \mathbf{A}_k\mathbf{C}_k\mathbf{w} &= \mathbf{0} \\ \mathbf{0} &= \mathbf{0} \end{aligned} \tag{6.58}$$

Part 2. Suppose $\mathbf{x}$ satisties $\mathbf{A}\left(\mathbf{x} - \mathbf{x}_k^\star\right) = \mathbf{0}$, then

$$\begin{aligned} E_k(\mathbf{x}) &= \| \mathbf{A}_k\mathbf{x} - \mathbf{b}_k \|^2 \\ &= \| \mathbf{A}_k\mathbf{x} - \mathbf{A}_k\mathbf{x}_k^\star + \mathbf{A}_k\mathbf{x}_k^\star - \mathbf{b}_k \|^2 \\ &= \| \mathbf{A}_k \left(\mathbf{x} - \mathbf{x}_k^\star\right) + \mathbf{A}_k\mathbf{x}_k^\star - \mathbf{b}_k \|^2 \\ &= \| \mathbf{A}_k\mathbf{x}_k^\star - \mathbf{b}_k \|^2 \end{aligned} \tag{6.59}$$

□

Substituting (6.56) into (6.39) yields:

$$\begin{aligned} h_i = &\min_{\mathbf{x}} E_i(\mathbf{x}) \\ \text{subject to } &\mathbf{A}_k(\mathbf{x} - \mathbf{x}_k^*) = 0 \; \forall k < i \\ &C(\mathbf{x}) = 0 \\ &\mathbf{D}\mathbf{x} + \mathbf{f} \geq 0. \end{aligned} \tag{6.60}$$

This can be solved recursively using N Quadratic Programs. This approach applies an increasing number of linear constraints at each step, which often lead to numerical instability. Kanoun and colleagues [2009] also describe a method for handling prioritized inequality constraints, which we do not discuss here. In practice, we have noticed this method has difficulties with rapidly changing constraints, such as when the number of contact points changes; noisy torques are generated often resulting in simulation failure. This formulation is also sensitive to the choice of QP solver, failing on active-set methods.

6.4 Summary

We have presented two algorithms for lexicographic optimization [Marler and Arora 2004] for the special case of quadratic objectives and linear constraints. We refer to these algorithms as unconstrained and constrained prioritized optimization.

Unconstrained prioritized optimization applies to problems with quadratic objectives and equality constraints. For this particular case, we propose a recursive algorithm based on solving a nested set of underconstrained linear systems. At each priority level, the linear system is solved in the nullspace basis of all previous solutions. Hence, results are guaranteed to not interfere with one another. A top-level objective is used to enforce inequality constraints. At each priority level, we perform a singular value decomposition (SVD). From this SVD we extract both the nullspace basis and calculate the pseudoinverse needed to solve the linear system.

Constrained prioritized optimization applies to problems with quadratic objectives and both equality and inequality constraints. To handle this case, we extend the unconstrained prioritized optimization algorithm. At each level we formulate a quadratic program (QP) to ensure that inequality constraints are respected; Inequality constraints are projected as needed.

When all objectives are combined at the same priority level, these methods are equivalent to weighted multiobjective optimization.

Chapter 7

Conclusion

Domo arigato Mr. Roboto

Styx, 1983

This thesis set out to develop new tools for designing and implementing control of simulated physics-based characters. Physics-based animation approaches hold the promise of improved generalization and compact representation, while also potentially shedding light on motor control processes used by animals.

We focus on locomotion actions, such as walking, jumping, and running, since they represent fundamental human behaviors. As such, control must deal with challenging aspects common to many types of human motion, including balance, intermittent contact, and underactuation; many practical applications stand to benefit from development of robust locomotion controllers.

Instead of parameterizing locomotion controllers in terms of joint behaviors, which implicitly couple control actions to character anthropometry, we focus on formulating locomotion using abstract features of pose. This requires three questions to be answered:

- What is a sufficient set of features needed to produce locomotion?
- How can individual features be regulated?
- How can multiple features be combined to design robust controllers?

Chapters 3 and 4 introduce feature-based control, an optimization-based approach that formulates control in terms of abstract high-level features. Control of each feature is formulated as a quadratic objective. We show that in-place actions, such as balancing, can be described in terms

of foot contact and momentum regulation objectives. More complex behaviors, such as jumping and walking, can be generated using task-specific state-machines, to coordinate time-varying features, such as foot plants and center-of-mass motion. We also propose objectives for regulating stylistic aspects of motion. This approach yields a variety of locomotion gaits with many human-like characteristics. Additionally, since control uses a model of the character dynamics, no additional effort is required to reuse control on new characters of different proportions or with a different number of joints.

To improve controller generalization and robustness, Chapter 5 presents a method for automatically selecting features targets. This method performs trajectory optimization using a low dimensional biomechanically-motivated planning model. The planning model uses linearized spring-loaded-inverted-pendulum dynamics in stance, projectile dynamics in flight, and a fixed two-footstep schedule. To select optimal planning model parameters, a nonlinear optimization is formulated that favors motion with user-specified characteristics (e.g., step length/duration, heading, etc.). This approach has several advantages. Firstly, by using a general mechanism to determine high-level features, hand-designed state machines, described in Chapter 4, can be completely eliminated. The resulting controller is memoryless: no persistent state (e.g., index into a state machine or reference trajectory) is needed by the controller, with the exception of remembering the last stance foot during flight. Secondly, using a planning horizon based on a specific number of foot steps, rather than on a fixed amount of time, we avoid special treatment of different behaviors (e.g., based on frequency domain motion characteristics). Hence, a single controller can be used to generate a large family of behaviors and transitions. Lastly, careful choice of objectives allows us to optimize essential features gait, producing simulated characters that can be directed at interactive rates.

A key element of the feature-based representation is a method for calculating joint-torques from feature objectives. To do this, we derive novel algorithms for problems with positive semidefinite quadratic objectives and linear constraints. These algorithms can be viewed as specializations of lexicographic optimization for quadratic programs. Our approach draws inspiration from two areas: optimization-based control [Abe et al. 2007; da Silva et al. 2008b; Hofmann et al. 2004] and prioritized task-space control [Khatib 1987; Khatib et al. 2004; Abe and Popović 2006]. The proposed algorithms guarantee minimal interference between objectives and do not require specialized projection matrices, as in Operational Space Control [Khatib et al. 2004].

7.1 Future Work

> When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one which has opened for us.
>
> Alexander Graham Bell, 1847-1922

The techniques described in this thesis provide a set of design tools for expressing motion in terms of features, objectives, and priorities, while abstracting out the specifics of individual joints or masses. This provides a poweful abstraction layer for control design that could be used for many different types of motion, not just locomotion.

A long-term goal is to build-up the vocabulary of features and associated control rules needed to generate the entire spectrum of human behaviors. For example, what features are required to generate a spinning karate-kick? What about a ballerina's fouetté? How can a run be modified to produce hurdle jumps? Can these changes be further exaggerated to generate a grand-jeté? Identifying common features and control mechanisms for various behaviors will lead to compact reusable motion models and, potentially, to a deepened understanding of human motion.

To date, we have found it challenging to generate certain types of motion with the described planning model and feature objectives. Using the preview model described in Chapter 5, we are unable to reliably generate sharp 180° turns when the character is running quickly. It is also challenging to hand-design controllers for classes of motion requiring substantial inertial changes in the sagittal plane, such as front/back flips. This arises since the body is assumed to be a point mass and rotational effects are neglected. To enforce this assumption, our AM objective actively cancels rotations about the center-of-mass. One potential solution is to explicitly account for coupling between linear and angular momentum [Macchietto et al. 2009]. Alternatively, our method might benefit from more sophisticated models of AM generation [Pratt et al. 2006; Lee and Goswami 2007; Orin and Goswami 2008; Lee and Goswami 2009]. This will require identifying appropriate features and objectives for regulating rotational quantities.

Additionally, it should be possible to generate a broader range of motions, by considering more sophisticated preview models. We currently use a fixed planning schedule for all generated motions, with forced left/right-foot alternation. Other motions, such as hopping on one leg could be achieved by enforcing left-foot-only contacts; more general contact sequences could be optimized, as in Wampler and Popović [2009].

The two foot-step planning horizon we currently use is also too short for certain terrain types. We find that the character has difficulty finding good footholds across gaps larger than 1 m and making anticipatory movements to walk on narrow beams. Toe-stubbing can occur when climbing up stairs larger than 0.5 m, since the current swing foot objective does not account for terrain collisions. Our method might benefit from "terrain-aware" swing strategies as in [Wu and Popović 2010]. Since we do not monitor inter-limb collisions, limb interpenetration can also occur.

The focus of the work in Chapter 5 is on robustness in challenging tasks and there is room for improvement in the style of motion. There are two main factors limiting the style. First, the SLIP model does not have knees, arms, and other elements that may affect style. Second, all of our examples err on the side of being overly stiff at all times. Although steady walking can be accomplished with fairly loose control, and strategies in Chapter 4 were designed with that in mind, the types of aggressive motions shown in this work require stiffer response. For example, if the arms are too loose, they can swing wildly during sharp turns. Varying full-body controller stiffness based on dynamic state or preview model predictions, would be an interesting direction for future work.

While we find that prioritized optimization is an important tool for identifying feature objectives and hand-designing control (Chapter 4), we did not find it to be essential for the motions in Chapter 5. One hypothesis is that frequent replanning, with a physically-consistent model, automatically corrects for deviations that may accrue when combining multiple, possibly conflicting, objectives by weighted combination.

Understanding the implications of different schedules and prioritization strategies on performance, solution convergence, temporal consistency, and motion style is an important next step for real-world use of our approach. For robotics applications, that may not have access to exact physical system parameters, combining feature-based techniques with system identification methods is also a potentially fruitful direction for future investigations.

Appendix A

Target Objective Derivation

A.1 Analytic Solution

Given x_0, $\dot{x}_0$, x_T, $\dot{x}_T$, we seek to find constants a, b, such that:

$$\begin{aligned}\ddot{x}(t) &= \left(1 - \frac{t}{T}\right) a + \frac{t}{T} b \\ &= \frac{b-a}{T} t + a. \end{aligned} \tag{A.1}$$

Integrating $\ddot{x}(t)$, we obtain:

$$\dot{x}(t) = \frac{b-a}{2T} t^2 + at + \dot{x}_0 \tag{A.2}$$

$$x(t) = \frac{b-a}{6T} t^3 + \frac{a}{2} t^2 + \dot{x}_0 t + x_0 \tag{A.3}$$

To solve for a and b, we substitute boundary conditions $\dot{x}(T) = \dot{x}_T$ and $x(T) = x_T$ into Equa-

tions A.2 and A.3.

$$\dot{x}_T = \frac{b-a}{2T}T^2 + aT + \dot{x}_0 \tag{A.4}$$
$$= \frac{b-a}{2}T + aT + \dot{x}_0 \tag{A.5}$$
$$= \frac{bT}{2} + \frac{aT}{2} + \dot{x}_0 \tag{A.6}$$
$$\frac{bT}{2} = \dot{x}_T - \dot{x}_0 - \frac{aT}{2} \tag{A.7}$$
$$b = \frac{2}{T}\left(\dot{x}_T - \dot{x}_0\right) - a \tag{A.8}$$

and

$$x_T = \frac{b-a}{6T}T^3 + \frac{aT^2}{2} + \dot{x}_0 T + x_0 \tag{A.9}$$
$$= \frac{b-a}{6}T^2 + \frac{aT^2}{2} + \dot{x}_0 T + x_0 \tag{A.10}$$
$$= \frac{aT^2}{3} + \frac{bT^2}{6} + \dot{x}_0 T + x_0 \tag{A.11}$$
$$a = \frac{3}{T^2}\left(x_T - x_0 - \dot{x}_0 T - \frac{bT^2}{6}\right) \tag{A.12}$$
$$= \frac{3\left(x_T - x_0 - \dot{x}_0 T\right)}{T^2} - \frac{b}{2} \tag{A.13}$$

Substituting Equation A.8 into Equation A.13:

$$a = \frac{3\left(x_T - x_0 - \dot{x}_0 T\right)}{T^2} - \frac{1}{2}\left(\frac{2}{T}\left(\dot{x}_T - \dot{x}_0\right) - a\right) \tag{A.14}$$
$$= \frac{3\left(x_T - x_0 - \dot{x}_0 T\right) - \left(\dot{x}_T - \dot{x}_0\right)T}{T^2} - \frac{a}{2} \tag{A.15}$$
$$= \frac{6\left(x_T - x_0\right) + 2\left(\dot{x}_T - 2\dot{x}_0\right)T}{T^2} \tag{A.16}$$

A.2 Least-Square Solution

Because $a = b = \lim_{T\to 0} = \infty$, simulation instabilities can arise if the analytic solution is used for small values of T. To avoid such instabilities, we solve for a and b in least-square sense. To

do this, we rewrite Equations A.13 and A.8 as:

$$T^2\left(2a+b\right) = 6(x_T - \dot{x}_0 T - x_0) \tag{A.17}$$

$$\frac{T}{2}\left(b+a\right) = \dot{x}_T - \dot{x}_0 \tag{A.18}$$

respectively. Expressing the above in matrix form yields:

$$\begin{bmatrix} T^2/3 & T^2/6 \\ T/2 & T/2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x_T - x_0 - \dot{x}_0 T \\ \dot{x}_T - \dot{x}_0 \end{bmatrix}. \tag{A.19}$$

To solve in a least-square sense Equation A.19 can be augmented with a user-specified tolerance ϵ as :

$$\underbrace{\begin{bmatrix} T^2/3 & T^2/6 \\ T/2 & T/2 \\ \epsilon & 0 \\ 0 & \epsilon \end{bmatrix}}_{A} \begin{bmatrix} a \\ b \end{bmatrix} = \underbrace{\begin{bmatrix} x_T - x_0 - \dot{x}_0 T \\ \dot{x}_T - \dot{x}_0 \\ 0 \\ 0 \end{bmatrix}}_{b} \tag{A.20}$$

and solved using the Moore-Penrose pseudoinverse:

$$x = (A^T A)^{-1} A^T b. \tag{A.21}$$

Alternatively Equation A.19 can be solved directly using a SVD-based singularity-robust pseudoinverse.

A.3 Equivalence to Cubic Hermite Interpolation

Hermite interpolation is a method of interpolating data points using polynomial functions that match data points in both value and first derivative. Given n values and n first derivatives Hermite interpolation computes a polynomial of at most degree $2n-1$ interpolating provided data.

In this section we show that for values of $T \gg 0$, the target objective derivation presented earlier is equivalent to cubic Hermite interpolation.

Proof. Given x_0, $\dot{x}_0$, x_T, $\dot{x}_T$, we seek to find constants α, β, γ, ψ, such that:

$$x(t) = \alpha t^3 + \beta t^2 + \gamma t + \psi \tag{A.22}$$

$$\dot{x}(t) = 3\alpha t^2 + 2\beta t + \gamma \tag{A.23}$$

$$\ddot{x}(t) = 6\alpha t + 2\beta \tag{A.24}$$

Substituting initial conditions $x(0) = x_0$ and $\dot{x}(0) = \dot{x}_0$ it can be shown that $\psi = x_0$ and $\gamma = \dot{x}_0$. Evaluating Equations A.22 and A.23 at $t = T$:

$$T^3\alpha + T^2\beta = x_T - x_0 - \dot{x}_0 T \tag{A.25}$$

$$3T^2\alpha + 2T\beta = \dot{x}_T - \dot{x}_0. \tag{A.26}$$

Comparing Equations A.24 and A.1 we set $2\beta = a$ and $6\alpha = (b-a)/T$, substitute them into Equation A.25 and expand :

$$T^3\frac{b-a}{6T} + \frac{T^2}{2}a = x_T - x_0 - \dot{x}_0 T \tag{A.27}$$

$$\frac{T^2}{2}a - \frac{T^2}{6}a + \frac{T^2}{6}b = \tag{A.28}$$

$$\frac{T^2}{3}a + \frac{T^2}{6}b = \tag{A.29}$$

Repeating this for Equation A.26:

$$\frac{2T}{2}a + 3T^2\frac{b-a}{6T} = \dot{x}_T - \dot{x}_0 \tag{A.30}$$

$$\frac{2T}{2}a - \frac{T}{2}a + \frac{3T^2}{6T}b = \tag{A.31}$$

$$\frac{T}{2}a + \frac{T}{2}b = \tag{A.32}$$

Rearranging Equations A.29 and A.32 in matrix form yields the original linear system from Equation A.19:

$$\begin{bmatrix} T^2/3 & T^2/6 \\ T/2 & T/2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x_T - x_0 - \dot{x}_0 T \\ \dot{x}_T - \dot{x}_0 \end{bmatrix} \tag{A.33}$$

□

Appendix B

Jacobian Computation

A key requirement of techniques that formulate motion control in terms of abstract goals, such as Feature-based Control, Virtual Model Control [Pratt 1995; Pratt 2000; Pratt et al. 2001; Coros et al. 2010], Operational Space Control [Khatib 1987], and inverse kinematics, is efficient Jacobian computation. All of these techniques require the Jacobian matrix:

$$\mathbf{J}^i = \frac{\partial \mathbf{p}_i}{\partial \mathbf{q}} \tag{B.1}$$

relating changes in joint angles, $\mathbf{q}$, to changes in position and orientation of some point $\mathbf{p}$ on a link i. Constraint-based formulations of impact and resting contact, used by forward dynamics simulators, also require these matrices.

Although the material in this appendix is not novel, treatment of this topic is currently spread across many sources in the literature. Additionally many references restrict treatment to special cases; analytical solutions are often presented for simple toy systems, while others focus on specialized joint types (e.g., one degree-of-freedom prismatic and revolute). This appendix aims to provide a concise summary of numerical techniques for calculating (B.1).

The appendix is organized as follows. First, we provide a formal definition of the Jacobian we seek to calculate. Second, we present various methods for calculating these Jacobians, focusing on techniques that evaluate the Jacobian at the origin of each link's local reference frame. Following the terminology in Featherstone [2008], we call this matrix the *Body Jacobian*. Third, we show that multiple Body Jacobians can be computed simultaneously. since all Jacobians with a shared subset of joints have duplicate columns. Lastly, we show how to transform Body Jacobians to arbitrary link locations.

B.1 Definition

Assume a non-linear function:

$$\mathbf{y} = f(\mathbf{q}) \tag{B.2}$$

exists mapping vectors $\mathbf{q} \in \mathbb{R}^n$ to vectors $\mathbf{y} \in \mathbb{R}^m$:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{B.3}$$

where $m \leq n$, and $\mathbf{y}$ and $\mathbf{q}$ depend on time. Differentiating both sides of (B.2) with respect to t, and applying the chain rule yields:

$$\frac{\partial \mathbf{y}}{\partial t} = \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t}. \tag{B.4}$$

This can be rewritten in the more familiar forms:

$$\dot{\mathbf{y}} = \mathbf{J}\dot{\mathbf{q}} \tag{B.5}$$

or

$$\delta\mathbf{y} = \mathbf{J}\delta\mathbf{q} \tag{B.6}$$

where,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial y_1}{\partial q_1} & \cdots & \frac{\partial y_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial q_1} & \cdots & \frac{\partial y_m}{\partial q_n} \end{bmatrix} \tag{B.7}$$

is the multidimensional form of the derivative, referred to as the Jacobian, and $\dot{\mathbf{y}}/\dot{\mathbf{q}}$ are the time-derivatives of $\mathbf{y}/\mathbf{q}$ respectively.

In character animation and robotics a Jacobian of common interest is (B.1). This corresponds to

the case:

$$\mathbf{y} = \mathbf{p}_i \tag{B.8}$$

where

$$\mathbf{p}_i = \begin{bmatrix} \mathbf{p}_i^a \\ \mathbf{p}_i^l \end{bmatrix} \tag{B.9}$$

is the angular and linear position of some point $\mathbf{p}$ on a link i. In the sections that follow, we drop the subscript on $\mathbf{p}_i$ for convenience. As in Section 3.1, n_l denotes the number of character links, n_j denotes the number of character joints, n_p denotes the dimensional of the generalized coordinate vector $\mathbf{q}$, and n_v denotes the dimensionality of the generalized coordinate velocity vector $\dot{\mathbf{q}}$.

B.2 Methods for Computing Jacobians

B.2.1 Finite-Difference Method

The simplest technique for estimating (B.1) involves extending the forward finite-difference approximation of the derivate:

$$\frac{dy}{dx} \approx \frac{y(x+\epsilon) - y(x)}{\epsilon} \tag{B.10}$$

to the multidimensional case. This is achieved by perturbing every component of $\mathbf{q}$ by some small amount ϵ and evaluating $\mathbf{p}$ multiple times. Each evaluation of $\mathbf{p}$ yields a column of $\mathbf{J}$, which we denote as $\mathbf{J}_j$. Algorithm 3 summarizes this procedure; $\mathbf{e}_j$ is the standard basis vector (i.e., a column vector of 0's with 1 in the j^{th} row).

Algorithm 3: Jacobian Estimation via Finite-Difference. $\mathbf{e}_j \in \mathbb{R}^{n_p \times 1}$

```
p ← f(q)
for j = 1 to n_p do
    p_ε ← f(q + e_j ε)
    J_j ← (p_ε − p) / ε
end
return J^i
```

Approximating the Jacobian using the finite-difference method has a number of problems. Firstly, the accuracy of the approximation depends on the size of ϵ. Making ϵ too big yields large errors, while making ϵ too small can result in quantities falling below machine precision. This problem is particularly acute near mechanical singularities (e.g., when the character's knees are fully extended). In these situations, solutions are very sensitive to the magnitude of ϵ. Secondly, for certain joint types, such as spherical joints represented using quaternions, care must be taken to ensure that perturbations do not violate joint representation assumptions (e.g., unit length constraints for quaternions [Shoemake 1985]). Although this issue can be addressed by renormalizing quaternions, additional errors will be introduced in derivatives since perturbations applied in the numerator and denominator of line 4 will be of different magnitudes. Lastly, finite-difference approximation is inefficient, since it requires $n + 1$ computations of the entire character kinematic state, which often recalculates/caches numerous transformations.

B.2.2 Unit-Velocity Method

A straightforward variant of the finite-difference method, that avoids many of its difficulties, is the Unit-Velocity Method. This simple approach is based on Equation B.5. Assuming a procedure:

$$\dot{\mathbf{p}} = f_{vel}(\mathbf{q}, \dot{\mathbf{q}}) \tag{B.11}$$

exists that can compute the velocity $\dot{\mathbf{p}}$ of some point $\mathbf{p}$, as a function of some $\mathbf{q}$ and $\dot{\mathbf{q}}$, individual Jacobian columns, $\mathbf{J}_j$, can be evaluated by setting $\dot{\mathbf{q}} = \mathbf{e}_j$. Algorithm 4 summarizes this approach.

Algorithm 4: Jacobian Calculation via Unit-Velocity Method. $\mathbf{e}_j \in \mathbb{R}^{n_p \times 1}$

```
1 for j = 1 to n_v do
2 |   J_j ← f_vel(q, e_j)
3 end
4 return J^i
```

Since no ad-hoc perturbation parameter is needed for this method, calculated Jacobians are exact (to machine tolerance), rather than approximate. Furthermore, no special treatment of different joints is required. Although spherical joints use quaternions to represent position, velocity is represented using an unconstrained angular velocity vector.

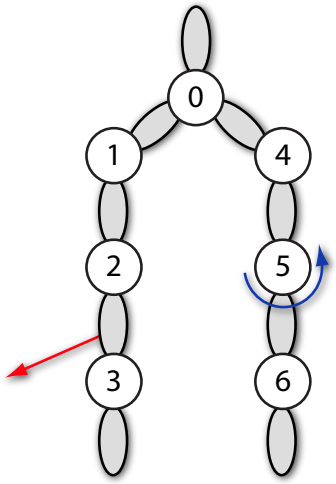

Figure B.1: Illustration of Transformation hierarchy. For this example $\boldsymbol{\lambda}_2 = [\,0,\ 1\,]$ (cf. Section B.2.3)

There are two problems with this method. First, it assumes that f_{vel} exists, which may not be the case. Assuming such a procedure is available, it must be evaluated n times. Second, it neglects that not all components of $\dot{\mathbf{q}}$ will contribute to velocities $\dot{\mathbf{p}}$. Specifically, for tree-like articulation hierarchies, such as characters, (B.1) will contain empty columns for all joints that do not influence $\dot{\mathbf{p}}$. Figure B.1 illustrates this idea using a schematic of the transformation hierarchy for the planar biped shown in Figure 3.4.

Let's assume the goal is to compute the Jacobian, $\mathbf{J}^2$, mapping joint velocities, $\dot{\mathbf{q}}$, to the Cartesian velocity at some location on link 2, illustrated by the red arrow in Figure B.1. Since link 2 is on a different branch of the transformation hierarchy than links/joints 4, 5, 6, there exists no change in the components of $\dot{\mathbf{q}}$ corresponding to these joints that will produce changes in $\dot{\mathbf{p}}$. Additionally, since link 3 is a child of link 2, movement in link 3 will also not induce any velocity in link 2. Hence, in this example, Jacobian columns corresponding to joints 2, 4, 5, 6 will always be $\mathbf{0}$.

B.2.3 Modified Unit-Velocity Method

To reduce unnecessary computation, Algorithm 4 can be extended to exploit knowledge of character topology. To accomplish this, arrays $\boldsymbol{\lambda}_i$, containing a list of parent links indices, are first computed for each of the character's links. As long as the character connectivity does not change,

these arrays can be cached and reused.

Once $\boldsymbol{\lambda}$ arrays are computed for all system joints, the unit-velocity method can be applied to each element of $\boldsymbol{\lambda}$. Algorithm 5 shows pseudocode for this method. For joints with more than one degree-of-freedom, care must be taken to set all corresponding columns in $\mathbf{J}$; a loop is provided in Algorithm 5 to handle this case.

Algorithm 5: Jacobian Calculation via Modified Unit-Velocity Method. $\mathbf{e}_j \in \mathbb{R}^{n_p \times 1}$

```
1  J^i ← 0
2  joints ← [ λ_i, i ]
3  for j in joints do
4  |   start ← getIndex( j )
5  |   ndofs ← getNumDofs( j )
6  |   for k = 1 to ndofs do
7  |   |   J_{start+k} ← f_vel(q, e_inx)
8  |   end
9  end
10 return J^i
```

B.3 Spatial-Axis Method

The discussion so far has assumed the existence of a function f_{vel} (cf. Equation B.11) that returns the linear and angular velocity of some arbitrary point on a specified link. In this section, we derive an efficient form of this function, by analysing the forward kinematic pass done on the transformation hierarchy in most animation systems. To focus the discussion, first we compute the *Body Jacobian* yielding velocities at the origin of each link's frame of reference. Following this, we show how Body Jacobians can be transformed to return the velocity at arbitrary link locations.

B.3.1 Body Jacobian

The velocity at some point on a link i:

$$\dot{\mathbf{p}}_i = \dot{\mathbf{p}}_{relative} + \dot{\mathbf{p}}_{parent} \tag{B.12}$$

is the sum of velocity link i's parent and the relative velocity between the link i and its parent. Since the velocity of link i's parent is similarly defined, this results in the recursive definition:

$$\dot{\mathbf{p}}_i = \dot{\mathbf{p}}_{relative} + \sum_{j=1}^{P} \dot{\mathbf{p}}_j \tag{B.13}$$

where P is the number of elements in $\boldsymbol{\lambda}_i$ (Section B.2.3).

Although there are many ways to represent relative link motion, a convenient approach is to use the spatial notation described by Feathersone [2008]. Using this approach, relative motion of an n degree-of-freedom joint, between parent and child links, is described using a *spatial axis* matrix $\mathbf{S}_i \in \mathbb{R}^{6\times n}$. For each $\mathbf{S}_i$, there is a corresponding vector $\dot{\mathbf{q}}_i \in \mathbb{R}^{n\times 1}$ that expresses the rate of change along each direction of $\mathbf{S}$. Stacked on top of one another, the set of $\mathbf{q}_i$ and $\dot{\mathbf{q}}_i$ fully specify the character's kinematic state (Section 3.1). Hence, Equation B.13 can be rewritten as:

$$\dot{\mathbf{p}}_i = \mathbf{S}_i\dot{\mathbf{q}}_i + \sum_{j=1}^{P} \dot{\mathbf{p}}_j. \tag{B.14}$$

Recall that the modified unit-velocity method iterates through all joints in the hierarchy that affect a particular link setting appropriate components of $\dot{\mathbf{q}} = \mathbf{e}_i$ (Algorithm 5; line 6). This calculates the relative contribution of each joint, while freezing all other joints (i.e., $\dot{\mathbf{p}}_j = 0$). Therefore, Equation B.14 simplifies to:

$$\dot{\mathbf{p}}_i = \mathbf{S}_i. \tag{B.15}$$

Hence, the inner loop of Algorithm 5 can be removed, yielding Algorithm 6.

Algorithm 6: Body Jacobian Calculation via Spatial-Axis Method

```
1 J^i ← 0
2 joints ← [ λ_i, i ]
3 for j in joints do
4 |   start ← getIndex( j )
5 |   ndofs ← getNumDofs( j )
6 |   J_{start:start+ndofs} ← _{world}X_{joint} S_{joint}
7 end
8 return J^i
```

Note that because each spatial axis, $\mathbf{S}_i$, is expressed in a link local coordinate system, columns of $\mathbf{J}$ must be transformed into a common coordinate frame; for convenience we choose the world reference frame. To perform this change of coordinates we used the 6×6 spatial transformation matrix:

$$ {}_b\mathbf{X}_a = ({}_b\mathbf{R}_a, \mathbf{r}) \tag{B.16} $$

$$ = \begin{bmatrix} {}_b\mathbf{R}_a & \mathbf{0} \\ {}_b\mathbf{R}_a \mathbf{r}\times^T & {}_b\mathbf{R}_a \end{bmatrix} \tag{B.17} $$

where ${}_b\mathbf{R}_a$ is the 3×3 homogeneous transform from frame a to frame b, and

$$ \mathbf{r}\times = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{B.18} $$

is the 3×3 skew symmetric matrix, and r is the offset between frame a and b expressed in frame a coordinates.

B.3.2 Jacobian at Arbitrary Link Location

Algorithm 6 returns the Body Jacobian. This is the Jacobian for the point at the origin of the link's coordinate frame. To calculate the Jacobian for other points $\mathbf{p}$, on a link we must apply an additional transformation:

$$ \mathbf{J}_i^{\mathbf{p}} = {}_{\mathbf{p}}\mathbf{X}_0 \mathbf{J}_i \tag{B.19} $$

where

$$ {}_{\mathbf{p}}\mathbf{X}_0 = (\mathbf{1}_{3\times3}, \mathbf{p}) \,. \tag{B.20} $$

Proof. We seek the velocity of some link i at some point $\mathbf{p}$. The velocity of any rigid body can be broken up into angular and linear terms, ω and $\mathbf{v}$ respectively. Whereas the angular velocity

is the same at all points on a body, the linear velocity varies through the body, yielding:

$$\dot{\mathbf{p}} = \begin{bmatrix} \omega \\ \omega \times \mathbf{p} + \mathbf{v} \end{bmatrix} \tag{B.21}$$

This is analogous to Equation B.12.

Starting from (B.19):

$$\begin{aligned}
\dot{\mathbf{p}} &= \mathbf{J}_i^{\mathbf{p}} \dot{\mathbf{q}} && \text{(B.22)} \\
&= {}_{\mathbf{p}}\mathbf{X}_0 \mathbf{J}_i \dot{\mathbf{q}} && \text{(B.23)} \\
&= \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{p}\times & \mathbf{1} \end{bmatrix} \mathbf{J}_i \dot{\mathbf{q}} && \text{(B.24)} \\
&= \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{p}\times & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{J}_i^{\omega} \\ \mathbf{J}_i^{v} \end{bmatrix} \dot{\mathbf{q}} && \text{(B.25)} \\
&= \begin{bmatrix} \mathbf{J}_i^{\omega} \\ -\mathbf{p} \times \mathbf{J}_i^{\omega} + \mathbf{J}_i^{v} \end{bmatrix} \dot{\mathbf{q}} && \text{(B.26)} \\
&= \begin{bmatrix} \omega \\ -\mathbf{p} \times \omega + \mathbf{v} \end{bmatrix} && \text{(B.27)} \\
&= \begin{bmatrix} \omega \\ \omega \times \mathbf{p} + \mathbf{v} \end{bmatrix} && \text{(B.28)}
\end{aligned}$$

□

References

Abe, Y., and Popović, J. 2006. Interactive Animation of Dynamic Manipulation. In *Proceedings of Symposium on Computer Animation (SCA)*, 195–204.

Abe, Y., da Silva, M., and Popović, J. 2007. Multiobjective Control with Frictional Contacts. In *Proceedings of Symposium on Computer Animation (SCA)*, 249–258.

Adamczyk, P. G., Collins, S. H., and Kuo, A. D. 2006. The advantages of a rolling foot in human walking. *Journal of Experimental Biology 209*, 20, 3953–3963.

Alexander, R. M. 1980. Optimum Walking Techniques for Quadrupeds and Bipeds. *Journal of Zoology 192*, 97–117.

Alexander, R. M. 1993. Optimization of Structure and Movement of the Legs of Animals. *Journal of Biomechanics 26*, 1, 1–6.

Allen, B., and Faloutsos, P. 2009. Complex Networks of Simple Neurons for Bipedal Locomotion. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*.

Allen, B., Chu, D., Shapiro, A., and Faloutsos, P. 2007. On the Beat! Timing and Tension for Dynamic Characters. In *Proceedings of Symposium on Computer Animation (SCA)*.

Atkeson, C., and Schaal, S. 1997. Robot Learning from Demonstration. In *Proceedings of the International Conference of Machine Learning (ICML)*, 12–20.

Auslander, J., Fukunaga, A., Partovi, H., Christensen, J., Hsu, L., Reiss, P., Shuman, A., Marks, J., and Ngo, J. T. 1995. Further Experience with Controller-based Automatic Motion synthesis for Articulated Figures. *Proceedings of SIGGRAPH 14*, 4, 311–336.

Azevedo, C., Poignet, P., and Espiau, B. 2002. Moving horizon control for biped robots without reference trajectory. In *Proceedings of International Conference on Robotics and Automation*, 2762–2767.

Baerlocher, P., and Boulic, R. 2004. An Inverse Kinematics Architecture Enforcing an Arbitrary Number of Strict Priority Levels. *The Visual Computer 20*, 6, 402–417.

Baraff, D. 1994. Fast Contact Force Computation for Nonpenetrating Rigid Bodies. In *Proceedings of SIGGRAPH*, 23–34.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press, March.

Bramble, D. M., and Lieberman, D. E. 2004. Endurance Running and the Evolution of Homo. *Nature 432*, 345–352.

Brotman, L. S., and Netravali, A. N. 1988. Motion Interpolation by Optimal Control. In *Proceedings of SIGGRAPH*, 309–315.

Byl, K., and Tedrake, R. 2008. Approximate Optimal Control of the Compass Gait on Rough Terrain. In *Proceedings of International Conference on Robotics and Automation (ICRA)*.

Byl, K., and Tedrake, R. 2008. Metastable Walking Machines. *International Journal of Robotics Research 28*, 8, 1040–1064.

Chestnutt, J. 2007. *Navigation Planning for Legged Robots*. PhD thesis, Carnegie Mellon University.

Clark, V. H., Chen, Y., Wilkens, J., Alaly, J. R., Zakaryan, K., and Deasy, J. O. 2008. Imrt Treatment Planning for Prostate Cancer using Prioritized Prescription Optimization and Mean-Tail-Dose functions. *Linear Algebra Applications 428*, 5-6, 1345–1364.

Cline, M. B., and Pai, D. K. 2003. Post-Stabilization for Rigid Body Simulation with Contact and Constraints. In *Proceedings of International Conference on Robotics and Automation*, vol. 3, 3744 – 3751.

Cohen, M. F. 1992. Interactive Spacetime Control for Animation. In *Proceedings of SIGGRAPH*.

Collins, S., Ruina, A., Tedrake, R., and Wisse, M. 2005. Efficient Bipedal Robots Based on Passive Dynamic Walkers. *Science 307*, 1082–1085.

Coros, S., Beaudoin, P., and van de Panne, M. 2009. Robust Task-based Control Policies for Physics-based Characters. *ACM Transactions on Graphics 28*, 5, 170.

Coros, S., Beaudoin, P., Yin, K., and van de Panne, M. 2010. Generalized biped walking control. *ACM Transactions on Graphics 29*, 3, –.

Craig, J. J. 1989. *Introduction to Robotics: Mechanics and Control*, 2nd ed. Addison-Wesley Longman Publishing Co.

da Silva, M., Abe, Y., and Popović, J. 2008. Interactive Simulation of Stylized Human Locomotion. *ACM Transactions on Graphics 27*, 3, 82.

da Silva, M., Abe, Y., and Popović, J. 2008. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. *Computer Graphics Forum 27*, 2, 371–380.

da Silva, M., Durand, F., and Popović, J. 2009. Linear Bellman Combination for Control of Character Animation. *ACM Transactions on Graphics 28*, 3, 1–1.

de Lasa, M., and Hertzmann, A. 2009. Prioritized Optimization for Task-Space Control. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*.

de Lasa, M., Mordatch, I., and Hertzmann, A. 2010. Feature-Based Locomotion Controllers. *ACM Transactions on Graphics 29*, 3.

Dickinson, M. H., Farley, C. T., Full, R. J., Koehl, M. A. R., Kram, R., and Lehman, S. 2000. How Animals Move: An Integrative View. *Science 288*, 5462 (April), 100–106.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. 2001. Composable Controllers for Physics-based Character Animation. In *Proceedings of SIGGRAPH*, 251–260.

Fang, A. C., and Pollard, N. S. 2003. Efficient Synthesis of Physically Valid Human Motion. *ACM Transactions on Graphics*, 417–426.

Featherstone, R. 2008. *Rigid Body Dynamics Algorithms*. Springer-Verlag.

Fujimoto, Y., Obata, S., and Kawamura, A. 1998. Robust Biped Walking with Active Interaction Control between Foot and Ground. In *Proceedings of International Conference on Robotics and Automation*, 2030–2035.

Full, R. J., and Koditschek, D. E. 1999. Templates and Anchors: Neuromechanical Hypotheses of Legged Locomotion on Land. *Journal of Experimental Biology 202*, 3325–3332.

Grzeszczuk, R., and Terzopoulos, D. 1995. Automated Learning of Muscle-Actuated Locomotion through Control Abstraction. In *Proceedings of SIGGRAPH*, 63–70.

Grzeszczuk, R., Terzopoulos, D., and Hinton, G. 1998. Neuroanimator: Fast Neural Network Emulation and Control of Physics-based Models. In *Proceedings of SIGGRAPH*, 9–20.

Guendelman, E., Bridson, R., and Fedkiw, R. 2003. Nonconvex Rigid Bodies with Stacking. *ACM Transactions on Graphics 22*, 3, 871–878.

Hansen, N. 2006. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. 75–102.

Harcourt-Smith, W. E. H., and Aiello, L. C. 2004. Fossils, Feet and the Evolution of Human Bipedal Loomotion. *Journal of Anatomy*, 403–416.

Hecker, C., Raabe, B., Enslow, R. W., DeWeese, J., Maynard, J., and van Prooijen, K. 2008. Real-time Motion Retargeting to Highly Varied User-Created Morphologies. *ACM Transactions on Graphics 27*, 3, 1–11.

Herr, H., and Popović, M. 2008. Angular Momentum in Human Walking. *Journal of Experimental Biology 211*, 467–481.

Hodgins, J. K., and Pollard, N. S. 1997. Adapting Simulated Behaviors for New Characters. In *Proceedings of SIGGRAPH*, 153–162.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. 1995. Animating Human Athletics. In *Proceedings of SIGGRAPH*, 71–78.

Hodgins, J. K. 1998. Animating Human Motion. *Scientific American* (March), 64–69.

Hofmann, A., Massaquoi, S., Popović, M., and Herr, H. 2004. A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. In *Proceedings of International Conference on Robotics and Intelligent Systems*.

Hsu, P., Mauser, J., and Sastry, S. 1989. Dynamic Control of Redundant Manipulators. *Journal of Robotic Systems 6*, 2, 133–148.

Isermann, H. 1982. Linear Lexicographic Optimimzation. *OR Spectrum 4*, 4, 223–228.

Jain, S., Ye, Y., and Liu, C. K. 2009. Optimization-Based Interactive Motion Synthesis. *ACM Transactions on Graphics 28*, 1, 1–10.

Kajita, S., and Tani, K. 1991. Study of Dynamic Biped Locomotion on Rugged Terrain - Derivation and Application of the Linear Inverted Pendulum Mode. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 1405–1411.

Kajita, S., Matsumoto, O., and Saigo, M. 2001. Real-time 3D Walking Pattern Generation for a Biped Robot with Telescopic Legs. In *Proceedings of International Conference on Robotics and Automation*, 2299–2306.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. 2003. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 1620–1626.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. 2003. Resolved Momentum Control: humanoid motion planning based on linear and angular momentum. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 1644–1650.

Kanoun, O., Lamiraux, F., Wieber, P.-B., Kanehiro, F., Yoshida, E., and Laumond, J.-P. 2009. Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 724–729.

Kawato, M. 1990. Feedback-Error-Learning Neural Network for Supervised Learning. *Advanced Neural Computers*, 365–372.

Khatib, O., Sentis, L., Park, J., and Warren, J. 2004. Whole Body Dynamic Behavior and Control of Human-Like Robots. *International Journal of Humanoid Robotics 1*, 1, 29–44.

Khatib, O. 1987. A Unified Approach to Motion and Force Control of Robot Manipulators: The Operational Space Formulation. *Journal of Robotics and Automation 3*, 1, 43–53.

Khatib, O. 1995. Inertial Properties in Robotics Manipulation: An Object-level Framework. *International Journal of Robotics Research 14*.

Kudoh, S., Komura, T., and Ikeuchi, K. 2006. Stepping Motion for a Human-like Character to Maintain Balance against Large Perturbations. In *Proceedings of International Conference on Robotics and Automation*, 2661–2666.

Kuffner, J., Nishikawa, K., Kagami, S., Inaba, M., and Inoue, H. 2003. Motion Planning for Humanoid Robots. In *Proceedings of International Symposium of Robotics Research (ISRR)*, 365–374.

Kuo, A. 1999. Stabilization of Lateral Motion in Passive Dynamics. *International Journal of Robotics Research 18*, 9, 917.

Kwon, T., and Hodgins, J. K. 2010. Control systems for Human Running using an Inverted Pendulum Model and a Reference Motion Capture Sequence. In *Proceedings Symposium on Computer Animation (SCA)*.

Laszlo, J., van de Panne, M., and Fiume, E. 1996. Limit Cycle Control and its Application to the Animation of Balancing and Walking. In *Proceedings of SIGGRAPH*, 155–162.

Laszlo, J., van de Panne, M., and Fiume, E. 2000. Interactive Control for Physically-based Animation. In *Proceedings of SIGGRAPH*.

Laszlo, J., Neff, M., and Singh, K. 2005. Predictive Feedback for Interactive Control of Physics-based Characters. In *Proceedings of Eurographics*.

Lee, S.-H., and Goswami, A. 2007. *Proceedings of International Conference on Robotics and Automation (ICRA)*. 4667–4672.

Lee, S.-H., and Goswami, A. 2009. *Humanoid Robots*. ch. The Reaction Mass Pendulum (RMP) Model for Humanoid Robot Gait and Balance Control, 166–178.

Liegeois, A. 1977. Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. *Transactions on Systems, Man and Cybernetics 7*, 12, 868–871.

Liu, C. K., and Popović, Z. 2002. Synthesis of Complex Dynamic Character Motion from Simple Animations. In *Proceedings of SIGGRAPH*.

Liu, Z., Gortler, S. J., and Cohen, M. F. 1994. Hierarchical Spacetime Control. In *Proceedings of SIGGRAPH*.

Liu, C. K., Hertzmann, A., and Popović, Z. 2005. Learning Physics-Based Motion Style with Nonlinear Inverse Optimization. *ACM Transactions on Graphics 24*, 3, 1071 – 1081.

Liu, L., Yin, K., van de Panne, M., Shao, T., and Xu, W. 2010. Sampling-Based Contact-Rich Motion Control. *ACM Transactions on Graphics 29*, 3, 10.

Loken, K. 2006. *Imitation-based Learning of Bipedal Walking Using Locally Weighted Regression*. Master's thesis, University of British Columbia, Computer Science Department.

Macchietto, A., Zordan, V., and Shelton, C. 2009. Momentum Control for Balance. *ACM Transactions on Graphics 28*, 3, 80.

Mansard, N., and Khatib, O. 2008. Continuous Control Law from Unilateral Constraints. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 3359–3364.

Marler, R. T., and Arora, J. S. 2004. Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization 26*, 6, 369–395.

McGeer, T. 1990. Passive Dynamic Walking. *International Journal of Robotics Research 9*, 2, 62–82.

McGeer, T. 1993. Dynamics and Control of Bipedal Locomotion. *Journal of Theoretical Biology 16*, 3, 277–314.

Miura, H., and Shimoyama, I. 1984. Dynamic Walk of a Biped. *International Journal of Robotics Research 3*, 2, 60–74.

Mordatch, I., de Lasa, M., and Hertzmann, A. 2010. Robust Physics-Based Locomotion Using Low-Dimensional Planning. *ACM Transactions on Graphics 29*, 3.

Morimoto, J., Cheng, G., Atkeson, C. G., and Zeglin, G. 2004. A Simple Reinforcement Learning Algorithm for Biped Walking. In *Proceedings of International Conference of Robotics and Automation*, 3030–3035.

MOSEK. MOSEK Optimization Software. `http://www.mosek.com`.

Muico, U., Lee, Y., Popović, J., and Popović, Z. 2009. Contact-Aware Nonlinear Control of Dynamic Characters. *ACM Transactions on Graphics 28*, 3, 81.

Nakamura, Y., Hanafusa, H., and Yoshikawa, T. 1987. Task-Priority Based Redundancy Control of Robot Manipulators. *International Journal of Robotics Research 6*, 2.

Nakanishi, J., and Schaal, S. 2004. Feedback Error Learning and Nonlinear Adaptive Control. *Neural Networks 17*, 1453–1465.

Nakanishi, J., Morimoto, J., Endo, G., Schaal, S., and Kawato, M. 2003. Learning from Demonstration and Adaptation of Biped Locomotion with Dynamical Movement Primitives. In *International Conference on Intelligent Robots and Systems*.

Nakanishi, J., Cory, R., Mistry, M., Peters, J., and Schaal, S. 2008. Operational Space Control: A Theoretical and Empirical Comparison. *International Journal of Robotics Research 27*, 6.

Neff, M., and Fiume, E. 2002. Modeling Tension and Relaxation for Computer Animation. In *Proceedings of Symposium on Computer Animation (SCA)*, 81–88.

Ngo, J. T., and Marks, J. 1993. Spacetime Constraints Revisited. In *Proceedings of SIGGRAPH*, 343–350.

Orin, D. E., and Goswami, A. 2008. Centroidal Momentum Matrix of a Humanoid Robot: Structure and Propreties. In *Proceedings of International Conference on Robotics and Intelligent Systems*.

Popović, M., Hofmann, A., and Herr, H. 2004. Angular Momentum Regulation during Human Walking: Biomechanics and Control. In *Proceedings of International Conference on Robotics and Automation*.

Popović, M. B., Goswami, A., and Herr, H. 2005. Ground Reference Points in Legged Locomotion: Definitions, Biological Trajectories and Control Implications. *The International Journal of Robotics Research 24*, 12, 1013–1032.

Pozzo, T., Berthoz, A., and Lefort, L. 1990. Head Stabilization During Various Locomotor Tasks in Humans. *Experimental Brain Research 82*, 97–106.

Pratt, J., and Pratt, G. 1998. Exploiting Natural Dynamics in the Control of a Planar Bipedal Walking Robot. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*.

Pratt, J., and Pratt, G. 1998. Intuitive Control of a Planar Bipedal Walking Robot. In *Proceedings of International Conference on Robotics and Automation*.

Pratt, J., and Tedrake, R. 2005. Velocity-Based Stability Margins for Fast Bipedal Walking. In *Fast Motions in Robotics and Biomechanics–Optimization and Feedback Control*, vol. 340, 299–324.

Pratt, J., Chew, C.-M., Torres, A., Dilworth, P., and Pratt, G. 2001. Virtual Model Control: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research*.

Pratt, J., Carff, J., Drakunov, S., and Goswami, A. 2006. Capture Point: A Step toward Humanoid Push Recovery. In *Proceedings Humanoid Robots*, 200–207.

Pratt, J. 1995. *Virtual Model Control of a Biped Walking Robot*. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Pratt, J. 2000. *Exploiting Inherent Robustness and Natural Dynamics in the Control of Bipedal Walking Robots*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 2007. *Numerical Recipes: The Art of Scientific Computing*, 3rd edition ed. Cambridge University Press.

Raibert, M. H., and Hodgins, J. K. 1991. Animation of Dynamic Legged Locomotion. *SIGGRAPH Computer Graphics 25*, 4, 349–358.

Raibert, M. H. 1986. *Legged Robots that Balance*. Massachusetts Institute of Technology, Cambridge, MA, USA.

Rebula, J., Canas, F., Pratt, J., and Goswami, A. 2007. Learning Capture Points for Humanoid Push Recovery. In *Proceedings of Humanoid Robots*, 65–72.

Rentmeester, M. J., Tsai, W. K., and Lin, K.-J. 1996. A Theory of Lexicographic Multi-Criteria Optimization. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, 76–79.

Ringrose, R. P. 1997. *Self-Stabilizing Running*. PhD thesis, Massachusetts Institute of Technology.

Safonova, A., Pollard, N. S., and Hodgins, J. K. 2003. Ootimizing Human Motion for the Control of a Humanoid Robot. In *Proccedings of International Symposium of Adaptive Motion of Animals and Machines (AMAM)*.

Safonova, A., Hodgins, J. K., and Pollard, N. S. 2004. Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces. In *Proceedings of SIGGRAPH*.

Schwind, W. J., and Koditschek, D. R. 2000. Approximating the Stance Map of a 2-DOF Monoped Runner. *Journal of Nonlinear Science*, 533–568.

Sentis, L. 2007. *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Stanford.

Sharon, D., and van de Panne, M. 2005. Synthesis of Controllers for Stylized Planar Bipedal Walking. In *Proceedings of International Conference on Robotics and Automation*, 2387–2392.

Shiratori, T., and Hodgins, J. K. 2008. Accelerometer-based User Interfaces for the Control of a Physically Simulated Character. In *ACM Transactions on Graphics*, vol. 27, 123.

Shiratori, T., Coley, B., Cham, R., and Hodgins, J. K. 2009. Simulating Balance Recovery Responses to Trips Based on Biomechanical Principles. In *Proceedings Symposium on Computer Animation (SCA)*.

Shkolnik, A., and Tedrake, R. 2008. High-Dimensional Underactuated Motion Planning via Task Space Control. *Proceedings of International Conference on Robotics and Intelligent Systems*, 3762–3768.

Shoemake, K. 1985. Animating rotation with quaternion curves. *SIGGRAPH Computer Graphics 19*, 3, 245–254.

Sims, K. 1994. Evolving Virtual Creatures. In *Proceedings of SIGGRAPH*, 15–22.

Smith, R. 1998. *Intelligent Motion Control with an Artificial Cerebellum*. PhD thesis, University of Auckland.

Sok, K. W., Kim, M., and Lee, J. 2007. Simulating Biped Behaviors from Human Motion Data. *ACM Transactions on Graphics 26*, 3, 107.

Spong, M. W. 1994. Partial Feedback Linearization of Underactuated Mechanical Systems. *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 314–321.

Srinivasan, M., and Ruina, A. 2006. Computer Optimization of a Minimal Biped Model Discovers Walking and Running. *Nature 439*, 7072, 72–75.

Stadler, W. 1988. *Multicriteria Optimization in Engineering and in the Sciences*. Plenum Press, ch. Fundamentals of Multicriteria Optimization, 1–25.

Sunada, C., Argaez, D., Dubowsky, S., and Mavroidis, C. 1994. A Coordinated Jacobian Transpose Control for Mobile Multi-Limbed Robotic Systems. In *Proceedings of International Conference on Robotics and Automation*, 1910–1915.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning*. MIT Press.

Takahashi, C. D., Scheidt, R. A., and Reinkensmeyer, D. J. 2001. Impedance Control and Internal Model Formation When Reaching in a Randomly Varying Dynamical Environment. *The Journal of Neurophysiology 86*, 2 (August), 1047–1051.

Talebinejad, S. 2000. *Compliant Running and Step Climbing of the Scout II Platform*. Master's thesis, McGill University.

Tedrake, R., and Seung, H. S. 2002. Improved dynamic stability using reinforcement learning. In *Proceedings International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, 341–348.

Tedrake, R., Zhang, T. W., and Seung, H. S. 2004. Stochastic Policy Gradient Reinforcement Learning on a Simple 3d Biped. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2849–2854.

Tedrake, R. 2009. LQR-trees: Feedback Motion Planning on Sparse Randomized Trees. In *Proceedings of Robotics: Science and Systems (RSS)*.

Thomson, W. T. 1993. *Theory of Vibration with Applications*, 4th ed. Prentice Hall.

Todorov, E., and Ghahramani, Z. 2003. Unsupervised Learning of Sensory-Motor Primitives. In *Proceedings International Conference of the IEEE Engineering in Medicine and Biology Society*.

Todorov, E., and Jordan, M. I. 2002. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience 5*, 11, 1226–1235.

Todorov, E., and Li, W. W. 2003. In *Proceedings of International Conference of the IEEE Engineering in Medicine and Biology Society*, 1758–1761.

Todorov, E. 2009. Efficient Computation of Optimal Actions. *Proceedings of The National Academy Of Sciences Of The United States Of America 106*, 38, 11478–11483.

Tsai, Y.-Y., Lin, W.-C., Cheng, K. B., Lee, J., and Lee, T.-Y. 2010. Real-Time Physics-Based 3D Biped Character Animation Using an Inverted Pendulum Model. *Transactions on Visualization and Computer Graphics 16*, 325–337.

van de Panne, M., and Fiume, E. 1993. Sensor-Actuator Networks. In *Proceedings of SIGGRAPH*, 335–342.

van de Panne, M., and Lamouret, A. 1995. Guided Optimization for Balanced Locomotion. In *Computer Animation and Simulation '95*.

van de Panne, M., Fiume, E., and Vranesic, Z. 1990. Reusable Motion Synthesis using State-Space Controllers. In *Proceedings of SIGGRAPH*, 225–234.

van de Panne, M., Kim, R., and Fiume, E. 1994. Virtual Wind-up Toys for Animation. In *Proceedings of Graphics Interface*.

Vijayakumar, S., D'souza, A., and Schaal, S. 2005. Incremental Online Learning in High Dimensions. *Neural Computation 17*, 12, 2602–2634.

Vukobratovic, M., and Juricic, D. 1969. Contribution to the Synthesis of Biped Gait. *IEEE Transactions on Bioemedical Engineering 16*, 1–6.

Wampler, K., and Popović, Z. 2009. Optimal Gait and Form for Animal Locomotion. *ACM Transactions on Graphics 28*, 3, 60.

Wang, J. M., Fleet, D. J., and Hertzmann, A. 2009. Optimizing Walking Controllers. *ACM Transactions on Graphics 28*, 5, 168.

Wang, J., Fleet, D. J., and Hertzmann, A. 2010. Optimizing Walking Controllers for Uncertain Inputs and Environments. *ACM Transactions on Graphics 29*, 3, 8.

Wills, A. QPC - Quadratic Programming in C. `http://sigpromu.org/quadprog`.

Winter, D. A. 2004. *Biomechanics and Motor Control of Human Movement*, 3rd ed. Wiley.

Witkin, A., and Kass, M. 1988. Spacetime Constraints. In *Proceedings of SIGGRAPH*, vol. 22, 159–168.

Wolpert, D. M., Ghahramani, Z., and Flanagan, J. R. 2001. Perspectives and Problems in Motor Learning. *Trends in Cognitive Science 5*, 11, 487–494.

Wooten, W. 1998. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology.

Wu, J. C., and Popović, Z. 2003. Realistic Modeling of Bird Flight Animations. In *Proceedings of SIGGRAPH*.

Wu, J.-C., and Popović, Z. 2010. Terrain-Adaptive Bipedal Locomotion Control. *ACM Transactions on Graphics 29*, 3, 10.

Wu, C.-C., and Zordan, V. 2010. Goal-Directed Stepping with Momentum Control. In *Proceedings Symposium on Computer Animation (SCA)*.

Yamane, K., and Hodgins, J. K. 2009. Simultaneous Tracking and Balancing of Humanoid Robots for Imitating Human Motion Capture Data. *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2510–2517.

Ye, Y., and Liu, C. K. 2010. Optimal Feedback Control for Character Animation using an Abstract Model. *ACM Transactions on Graphics 29*, 3, 8.

Yin, K., Loken, K., and van de Panne, M. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Transactions on Graphics 26*, 3, 81.

Yin, K., Coros, S., Beaudoin, P., and van de Panne, M. 2008. Continuation Methods for Adapting Simulated Skills. *ACM Transactions on Graphics 27*, 3.

Zhao, P., and van de Panne, M. 2005. User Interfaces for Interactive Control of Physics-Based 3D Characters. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*.

Zordan, V., and Hodgins, J. K. 2002. Motion Capture-Driven Simulations that Hit and React. In *Proceedings of Symposium on Computer Animation (SCA)*, 89–96.

Zykina, A. V. 2003. A Lexicographic Optimization Algorithm. *Automation and Remote Control 65*, 3, 363–368.