Data-Driven Kinematic and Dynamic Models for Character Animation

by

KangKang Yin

B.Sc., ZheJiang University, 1997

M.Sc., ZheJiang University, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

October 2007

Abstract

Human motion plays a key role in the production of films, video games, virtual reality applications, and the control of humanoid robots. Unfortunately, it is hard to generate high quality human motion for character animation either manually or algorithmically. As a result, approaches based on motion capture data have become a central focus of character animation research in recent years.

We observe three principal weaknesses in previous work using data-driven approaches for modelling human motion. First, basic balance behaviours and locomotion tasks are currently not well modelled. Second, the ability to produce high quality motion that is responsive to its environment is limited. Third, knowledge about human motor control is not well utilized. This thesis develops several techniques to generalize motion capture character animations to balance and respond. We focus on balance and locomotion tasks, with an emphasis on responding to disturbances, user interaction, and motor control integration.

For this purpose, we investigate both kinematic and dynamic models. Kinematic models are intuitive and fast to construct, but have narrow generality, and thus require more data. A novel performance-driven animation interface to a motion database is developed, which allows a user to use foot pressure to control an avatar to balance in place, punch, kick, and step. We also present a virtual avatar that can respond to pushes, with the aid of a motion database of push responses. Consideration is given to dynamics using motion selection and adaption.

Dynamic modelling using forward dynamics simulations requires solving difficult problems related to motor control, but permits wider generalization from given motion data. We first present a simple neuromuscular model that decomposes joint torques into feedforward and low-gain feedback components, and can deal with small perturbations that are assumed not to affect balance. To cope with large perturbations we develop explicit balance recovery strategies for a standing character that is pushed in any direction. Lastly, we present a simple continuous balance feedback mechanism that enables the control of a large variety of locomotion gaits for bipeds. Different locomotion tasks, including walking, running, and skipping, are constructed either manually or from motion capture examples. Feedforward torques can be learned from the feedback components, emulating a biological motor learning process that leads to more stable and natural motions with low gains. The results of this thesis demonstrate the potential of a new generation of more sophisticated kinematic and dynamic models of human motion.

Contents

Abstract . . . . . **ii**

Contents . . . . . **iii**

List of Figures . . . . . **vii**

List of Tables . . . . . **ix**

Acknowledgements . . . . . **x**

1 Introduction **1**
1.1 Motivation 1
1.2 Models for Character Animation 2
1.3 Thesis Organization 3

2 Related Work **9**
2.1 Kinematic Animation and Optimization 9
2.1.1 Motion Editing and Data-driven Animation 9
2.1.2 Trajectory Optimization 10
2.1.3 Animation Interfaces 13
2.2 Dynamic Simulation and Robotics 13
2.2.1 Balance Indices 14
2.2.2 Equations of Motion and Control 17
2.2.3 Learning from Humans 19
2.2.4 Dynamic Simulation for Animation 19
2.3 Motor Control 20
2.3.1 Biomechanics 20
2.3.2 Computational Motor Control 21
2.3.3 Postural and Balance Control Strategies 28

3 Data-Driven Kinematic Animation from Foot Pressure: FootSee **32**
3.1 Motivation 33
3.1.1 System Overview 35
3.2 Related Work 36

3.3 Data Capture and Processing . . . 36
3.3.1 Data Capture . . . 36
3.3.2 Feature Extraction . . . 38
3.4 Motion Recognition . . . 40
3.5 Motion Editing . . . 42
3.5.1 Inverse Kinematics for Stepping . . . 42
3.5.2 Foot-ground Contact Satisfaction . . . 43
3.5.3 Timewarping . . . 44
3.6 Results . . . 44
3.7 Discussion . . . 45
3.8 Future Work . . . 47

4 Data-Driven Kinematic Animation of Interactive Balancing . . . 48
4.1 Motivation and Related Work . . . 49
4.1.1 Biomechanics . . . 50
4.1.2 Robotics . . . 50
4.1.3 Computer animation . . . 51
4.2 Background . . . 53
4.2.1 Balance Strategies . . . 53
4.2.2 Momentum . . . 53
4.2.3 Push Impulse Parametrization . . . 54
4.3 Motion Capture and Database Construction . . . 54
4.4 Motion Selection . . . 56
4.5 Motion Adaptation . . . 58
4.5.1 Scaling . . . 58
4.5.2 Rotation . . . 59
4.5.3 Blending . . . 61
4.6 Results and Discussion . . . 62

5 Dynamic Animation of Small Perturbations . . . 65
5.1 Motivation for Motion Perturbation . . . 66
5.2 Related Work . . . 67
5.2.1 Physics-based Animation . . . 67
5.2.2 Biologically Based Motor Control . . . 68
5.3 Our Motor Control Model . . . 69
5.4 Dynamics Framework . . . 71
5.4.1 Equations of Motion . . . 71
5.4.2 Forward Dynamics . . . 72
5.4.3 Inverse Dynamics . . . 72
5.5 Integrating Motor Control with Dynamic Simulation . . . 73
5.5.1 Algorithm Summary . . . 73

5.5.2 Inverse Dynamics Preprocessing . . . 74
5.5.3 Combining Feedback and Feedforward Torques . . . 75
5.6 Results . . . 75
5.7 Discussion and Future Work . . . 76

6 Dynamic Animation of Interactive Balancing . . . 79
6.1 Related Work . . . 80
6.1.1 Common Points of Reference . . . 80
6.1.2 Data-Driven Approaches to Balance Control . . . 81
6.1.3 Balance Without Stepping . . . 81
6.1.4 Balance During Walking . . . 81
6.1.5 Stepping Response Models . . . 82
6.1.6 Overview of Human Balance Strategies . . . 83
6.2 In-place Controllers . . . 83
6.2.1 Ankle Strategy . . . 83
6.2.2 Hip Strategy . . . 84
6.3 Stepping Controllers . . . 85
6.3.1 Where and How to Step . . . 86
6.3.2 Single and Double Stepping . . . 88
6.3.3 Multi Stepping Controller . . . 89
6.4 Multiple Strategy Integration . . . 89
6.5 Results . . . 90
6.6 Discussion and Future Work . . . 93

7 Dynamic Animation of Locomotion: SIMBICON . . . 96
7.1 Overview . . . 97
7.1.1 Overview of Our Approach . . . 97
7.2 Related Work . . . 99
7.3 Balance Control Strategy . . . 100
7.3.1 Finite State Machine . . . 101
7.3.2 Torso and Swing-hip Control . . . 101
7.3.3 Balance Feedback . . . 102
7.4 Manual Controller Design . . . 103
7.5 Controllers from Motion Capture Data . . . 105
7.6 Feedback Error Learning . . . 106
7.7 Results . . . 108
7.7.1 2D Biped Locomotion . . . 109
7.7.2 3D Biped Locomotion . . . 112
7.7.3 Setting the Balance Feedback Gain Parameters . . . 119
7.7.4 Limitations . . . 119
7.8 Discussion . . . 120

8 Conclusions **123**
8.1 Contributions 123
8.2 Future Work 124

Appendix A More About Balance Indices **126**
A.1 GRF and CoP 126
A.2 ZMP 126
A.3 FRI 128

Appendix B Hardware used in Our Experiments **129**
B.1 Equipment 129
B.2 Synchronization between Vicon and XSensor 130
B.3 XSensor Setup 131

Appendix C Inverse Kinematics Algorithm for FootSee **132**
C.1 Derivation of α 135
C.2 Flexion of Knee 136

Appendix D Details of Our Dynamics Simulator **137**
D.1 Equations of Motion 137
D.2 Details of Matrices J and H 139
D.3 Forward Dynamics 139
D.3.1 Accommodating Stiffness 140

Bibliography **141**

List of Figures

1.1 Icons for modelling approach 4
1.2 Input-output diagram of Chapter 3 5
1.3 Input-output diagram of Chapter 4 5
1.4 Input-output diagram of Chapter 5 6
1.5 Input-output diagram of Chapter 6 7
1.6 Input-output diagram of Chapter 7 7

2.1 Motion capture in animation and biomechanics 10
2.2 Explanation of trajectory optimization 11
2.3 Humanoids and industrial robots 14
2.4 Ground reaction force 15
2.5 CoG, ZMP, and FRI 16
2.6 Articulated rigid body system 18
2.7 A Complete Motor System 22
2.8 Internal models 24

3.1 A preview of our results 33
3.2 Data-driven kinematic animation system 34
3.3 FootSee system block diagram 35
3.4 Custom XSensor pressure sensor pad. 37
3.5 Sample data for FootSee 38
3.6 Feature window for motion recognition 40
3.7 Weights for motion distance computation 41
3.8 Comparison of input and synthesized poses 46

4.1 Kinematic balance system block diagram 49
4.2 A preview of our results 49
4.3 Momentum of a balancing motion using arm-rotation strategy 52
4.4 Parametrization of push impulses 55
4.5 Linear momentum of a motion and its scaled motion 59
4.6 Angular momentum of a motion and its rotated motion 60
4.7 Calculation of momenta difference 60
4.8 Foot support polygon 61

4.9 Motion database 62
4.10 Balance behaviors upon perturbations 63

5.1 Muscle model 66
5.2 A preview of our results 66
5.3 Our reference model of the motor system 69
5.4 Skeleton used for motion perturbation 76
5.5 Comparison of an original and its perturbed arm motions 77
5.6 Comparison of original and perturbed full body motions 78

6.1 A preview of our results 80
6.2 In-place balance strategies 84
6.3 Pose control graph 87
6.4 Interpolation of control points for single stepping 88
6.5 Domains of different balance strategies 91
6.6 The domain nest of balance strategies 92
6.7 Foot configurations of push responses 94
6.8 Push responses using different strategies 95

7.1 A preview of our results 98
7.2 Finite state machine for walking 101
7.3 Elements of the balance control strategy 102
7.4 GUI for controller parameters 104
7.5 DoFs illustration of 2D and 3D models 109
7.6 Manually-designed controllers 111
7.7 Planar biped walking on rough terrain 112
7.8 FEL on 2D biped 113
7.9 Tai Chi cloud hands 113
7.10 Imitation of motion capture data 114
7.11 Variations in locomotion, illustrated using footprints 116
7.12 Climbing a slope 117
7.13 The largest pushes recoverable by a 3D walker. 118
7.14 GRF in simulation. 120

B.1 Experiment setup 129
B.2 Synchronization circuit 130

C.1 Joint angles plot of stepping 133
C.2 Simple analytical IK 134

List of Tables

1.1 Thesis organization 4

6.1 Summary of work dealing with push recovery 93

7.1 Parameters of various locomotion controllers 122

Acknowledgements

I would like to thank my supervisor Dinesh Pai for initially getting me interested in character animation. It has been a challenging and fascinating journey for me ever since. Dinesh's ideas, passion and energy for research keep amazing me everyday. I wish to thank my supervisor Michiel van de Panne. I had a blind belief that walking is not hard, and Michiel helped me realize this dream. I admire him for his enthusiasm, ideas, and insights. His patience, kindness and considerations for students have helped me through the most difficult time of my graduate student life. Michiel has become my role model as a researcher and professor. Without their encouragement and support, this thesis would never have been possible. I am grateful for having the karma to know and work with them both.

I want to thank my supervisory committee member Chen Greif for his kindness and support for this work, and making math classes fun. I would like to thank my external examiner and university examiners for their valuable suggestions to this thesis. I should thank my advisor Jim Little for his support when I first came to Canada. His great understanding of my Chinese English is much appreciated. I would like to thank faculty members Ian Mitchell and Michael Friedlander for their various help and discussions.

Special thanks go to my fellow graduate students Mike Cline, Paul Kry, Peng Zhao, and Kevin Loken. I had wonderful collaborations with them, and their contributions are key to this thesis. I would like to thank my friends Stelian Coros, Jesse Hoey, Danny Kaufman, Qiang Kong, Joshua Lee, Dan Li, Etienne Lyard, Huanchun Ren, Subarna Sadhukhan, Shinjiro Sueda, Zhijin Wang, Qi Wei, Chen Yang, Suwen Yang, Rong Yi, XiaoDong Zhou. They made my life at UBC memorable.

Finally, I want to thank my parents and sister, for always being there for me no matter what, for always believing in me no matter what. I cannot imagine a life without their love.

KangKang Yin

The University of British Columbia
October 2007

Chapter 1

Introduction

The primary goal of this thesis is the development of techniques for generalizing balanced character animation from motion capture examples. Both kinematic and dynamic models are investigated.

In this chapter, we first explain what motivates our thesis work, followed by a general discussion of character animation models, their strengths and weaknesses, and what our work aims to improve upon. Lastly we give an overview of the thesis organization.

1.1 Motivation

Character animation is widely used today in computer graphics film production, video games, and virtual reality applications. Artistic skills are heavily relied upon to manually produce good quality motions. However, it is an expensive and tedious approach, considering the large number of degrees of freedom (DoF) the artists have to control, the complexity of the coordinations both in time and in space among all the DoFs, and all the environmental and task-oriented constraints that a specific motion has to satisfy.

A more automated approach to character animation thus seems appealing, yet remains difficult. The complexities of human motor control and biomechanics are far from fully understood. It is also difficult because every human is an expert in observing human motion. We are very familiar with how our body moves and balances since we do it everyday. Even though most of us are not artists, we can still feel when something is wrong with an animation clip without always being able to identify exactly what is wrong.

Because motion capture directly records the subtleties of human motion, it has become popular as a means for creating animated human movement. With relative ease, one can place motion capture markers on humans, capture their motion, and in turn create compelling animation from the capture data. By comparison, the alternative of using artists and algorithms to produce convincing motion is considerably

more time consuming and difficult.

Motion capture is not without problems. Motion capture equipment is usually expensive. Motion capture sessions and data post-processing are time consuming and error prone. One cannot capture every conceivable motion. We thus want to reuse the good data as much as possible. A major challenge in motion capture research is that of generalizing recorded examples to satisfy the constraints of new animations, and to respond to new environmental perturbations. Being able to generalize motions observed from human subjects can also help our understanding of the biomechanics and control of balance. It may also provide valuable insights into the control of biped and humanoid robots.

With the goal of generalizing motion capture examples established, we need to decide on a subset of motions to focus on, given that the human machine is capable of performing so many types of movement. The opportunities for character animation research seem endless, ranging from animating fine object manipulations to spectacular stunts, and from animating small scale body parts to large scale crowds. Yet, we can note that common movements such as balance and locomotion, which are among the early skills that infants learn, are not very well solved. Foot-skate and abrupt transitions are common in kinematic approaches. For approaches based on simulating the forward dynamics, the control of biped balance and locomotion is known to be difficult. Bipeds are unstable, underactuated, high-dimensional dynamical systems. The motor control mechanisms of biped balance and locomotion are far from fully understood. Only a small number of groups in the world have demonstrated the successful control of dynamic humanoid walking either in simulation or in an actual robot. For both kinematic and dynamic approaches, models that support interactions of characters with their environment, such as pushes, are lacking.

In summary, we wish to generalize motion capture examples, focusing on balance and locomotion tasks, and interactive responses to the environment. We further wish to integrate motor control principles wherever possible.

1.2 Models for Character Animation

The problem of character animation can be approached from a number of different perspectives. Starting from motion capture data, there are usually two paths to carry on reusing the examples. If we care only about regenerating trajectories, without much concern about respecting the underlying physics, we can use kinematic models. On the other hand, if we want to use examples to help reconstruct a controller which outputs torques to a dynamic simulator or a robot, we need to use dynamic models coupled with appropriate control strategies.

Kinematic modelling is usually easier to implement, and can be quite effective in well-defined scenarios. However, due to the fundamental limitations of trajectory

manipulation, large data sets are needed to ensure the quality of newly synthesized trajectories. That is, kinematic modelling has an innate data-driven flavor. The underlying assumption is that when trajectories are similar, the underlying physics and motor control can be interchangeable. Trajectory editing, and blending techniques thus rely on the closeness of sample trajectories to be successful. Interpolation techniques are commonly used, while extrapolation is avoided. In this thesis, we develop kinematic modelling in two scenarios. (1) We use it to animate a virtual avatar using performance animation; (2) we use it to develop models of the balancing behaviors of a standing character responding to interactive pushes.

Kinematic modelling cannot, however, guarantee physical realism irrespective of how much data we gather. Its quality is fundamentally limited by the size of the backend motion database. Furthermore, kinematic methods provide little insight into underlying balance strategies and motor mechanisms. In contrast dynamic modelling, such as the use of physics-based simulations, can guarantee the physical plausibility. The fidelity of the simulation results is constrained by the details of the dynamic models, and fidelity of the dynamic parameters with respect to the parameters of the real world that we want to simulate.

Writing a physics-based dynamic simulator is not trivial. The real key to the success of dynamic modelling, however, is how to implement the control. Complex biomechanical, sensory, neural, and muscular subsystems are involved in human postural and balance control. The balance trajectories we capture are the results of complex interactions between environmental disturbances and constraints and complex human motor control mechanisms. Ideally, we would wish to reverse engineer "intelligent" controllers from the trajectories.

The aspects of dynamic modelling we develop in this thesis include: (1) the introduction of simple neuromuscular control models into dynamic simulations to model reactions to small perturbations; (2) the study of character balance behaviours to large pushes. (3) the construction of simple and robust balance and locomotion controllers based on a set of control laws and insights into motor control. While dynamic modelling still has many remaining challenges, we believe that it has significant potential for character animation in the future.

1.3 Thesis Organization

The section describes the organization and contributions of each chapter. Table 1.1 shows the organization of this thesis, with a coarse classification of modelling approach, i.e., kinematic or dynamic, and a short description of what problem each chapter addresses. There are approaches and problems that we do not explore in this thesis, and we label these cells as *Ai* in the domain table. We will discuss them in the related work and future work sections of this thesis.

kinematic	dynamic	approach / problem
Chapter 3	*A1*	performance animation
A2	*Chapter 5*	reactions to small perturbations
Chapter 4	*Chapter 6*	reactions to pushes while standing
A3	*Chapter 7*	balance during locomotion

Table 1.1: Thesis organization. Cells labelled *Ai* designate areas that are not explored in this thesis.

We also give a simple input-output diagram for each chapter. Arrows represent user-specified pushes or environmental perturbations. Filmstrips represent captured motions or synthesized animations. The type of computation, i.e., modelling approach, is represented using two icons as explained in Figure 1.1. The database icon with trajectories inside means that the system is of a data-driven kinematic nature. The equation icon implies that the system involves controller construction and forward dynamic simulation.

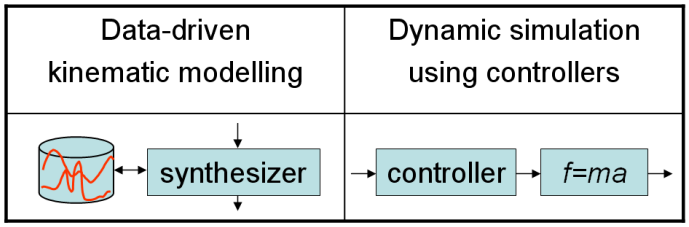


Figure 1.1: Icons for modelling approach.

Chapter 2 provides a general survey of related work in graphics and other fields of research, and reviews the background knowledge for later chapters. We assume the reader to have a basic knowledge of kinematic animation and dynamic simulation. Discussion and citations of the most important related work appear throughout the thesis.

Chapter 3 describes a kinematic animation system, named FootSee, which we develop using a combined motion capture and foot-ground interaction capture system. We present a simple-to-use animation interface that uses a foot pressure sensor pad to interactively control avatars for video games, virtual reality, and low-cost performance-driven animation. During an offline training phase, we capture full body

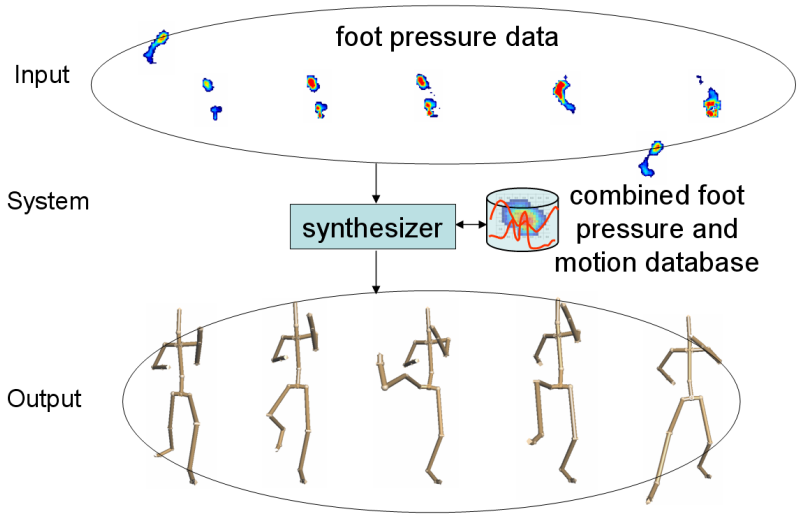


Figure 1.2: Input-output diagram of Chapter 3.

motions with a motion capture system, as well as the corresponding foot-ground pressure distributions with a pressure sensor pad, into a database. At run time, the user acts out the animation desired on the pressure sensor pad. The system then reconstructs a plausible motion using only observations of the foot-ground interactions. The most appropriate motions from the database are selected, and edited online to drive the avatar. We describe our motion recognition, motion blending, and inverse kinematics algorithms in detail. They are easy to implement, and cheap to compute. FootSee can control a virtual avatar with a fixed latency of one second with reasonable accuracy. Our system thus makes it possible to create some types of interactive animations without the cost or inconveniences of a full body motion capture system. Figure 1.2 shows the input and output of the system. This work is published as [Yin and Pai 2003]. The corresponding animation video can be downloaded at http://www.cs.ubc.ca/~kkyin/animation/Yin_SCA03.avi.

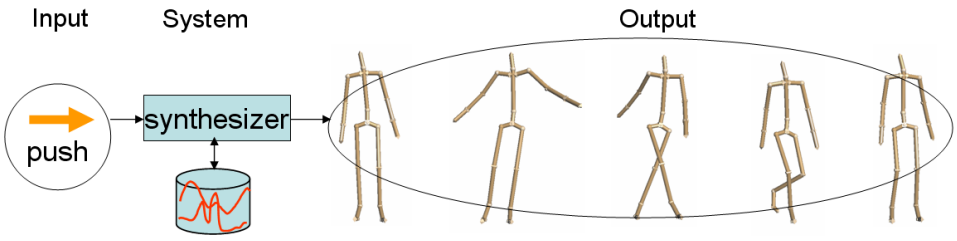


Figure 1.3: Input-output diagram of Chapter 4.

Chapter 4 continues with kinematic modelling, and describes a data-driven approach for producing interactive dynamic balancing behaviors for an animated character. The result is a standing character that can interactively respond to single

or multiple pushes in various directions and of varying magnitudes. A database of captured responses to pushes is used to create a model that supports hip, arm, and stepping strategies for balance recovery. An interactive push is modelled as a force impulse, which is then used to compute a momentum-based motion index in order to select the most appropriate recovery motion from the database. The selected motion is then adapted in order to provide a response that is specifically tailored to the given force impulse while preserving the realism and style of the original motion. Based on a relatively small motion database, our system is effective in generating various interactive balancing behaviors, for single and multiple pushes. Figure 1.3 shows the input and output of the system. This work is published as [Yin et al. 2005]. The corresponding animation video can be downloaded at http://www.cs.ubc.ca/∼kkyin/animation/Yin_PG05.wmv.

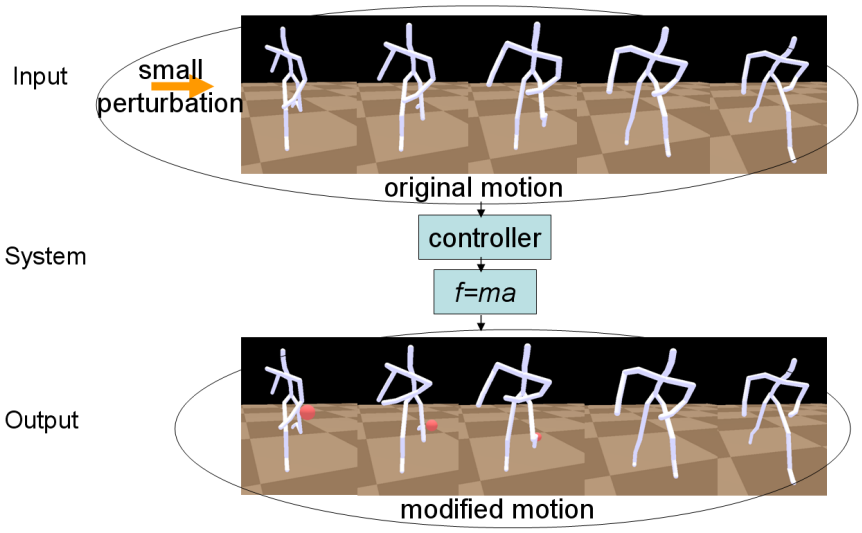


Figure 1.4: Input-output diagram of Chapter 5.

In Chapter 5 we shift to using dynamic modelling. We develop a dynamics-based framework that explicitly takes into account motor control by using a simple human neuromuscular control model. Our model of muscle forces includes a feedforward term and low gain feedback. The feedforward component is calculated from motion capture data using inverse dynamics. The feedback component generates reaction forces to unexpected small external disturbances. The perturbed animation is then resynthesized using forward dynamics. This allows us to create animations where the character reacts to unexpected external forces in a natural way, e.g., when the character is hit by a ball, but still retains qualities of the original animation. Such a technique is likely to be useful for applications such as interactive sports video games. Figure 1.4 shows the input and output of the system. This work is published as [Yin et al. 2003]. The corresponding animation video can be downloaded at http://www.cs.ubc.ca/∼kkyin/animation/Yin_PG03.avi.

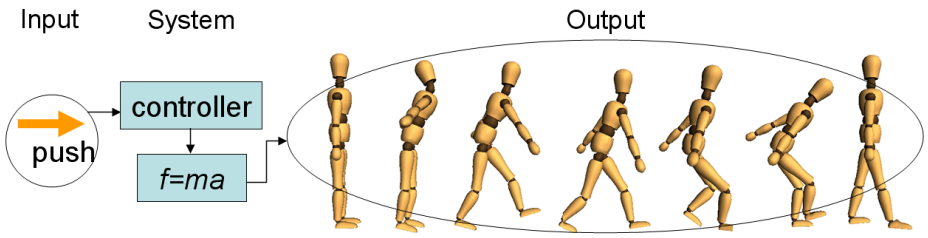


Figure 1.5: Input-output diagram of Chapter 6.

Chapter 6 parallels Chapter 4, only now using a dynamics-based approach. We develop, integrate, and evaluate humanoid balance controllers that can recover from unexpected external perturbations of varying magnitudes in arbitrary directions. The starting state is a normal standing pose having the feet placed with a shoulder width spacing. The supported balance strategies include ankle and hip strategies for in-place balance, single-step, double-step, and multi-step balance recovery. These strategies are further integrated with a limit cycle based walking controller. The limitations of each type of controller are mapped out in terms of the maximum push magnitudes and directions they can sustain. The controller construction can be informed using motion capture data if desired. Results are provided for a 30 DoF humanoid simulation that can be controlled at interactive rates. Figure 1.5 shows the input and output of the system. This work is not previously published. The corresponding animation video can be downloaded at http://www.cs.ubc.ca/~kkyin/animation/Yin_SUB07.mov.

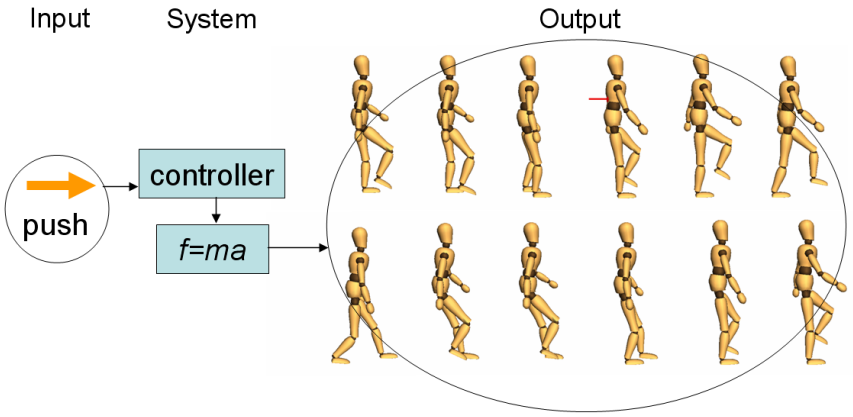


Figure 1.6: Input-output diagram of Chapter 7.

Chapter 7 presents SIMBICON, a simple control strategy that can dynamically generate a large variety of gaits and styles in real-time, including walking in all directions (forwards, backwards, sideways, turning), running, skipping, and hopping. Controllers can be authored using a small number of parameters, or their construction can be informed by motion capture data. The controllers are applied to 2D and

3D physically-simulated character models. Their robustness is demonstrated with respect to pushes in all directions, unexpected steps and slopes, and unexpected variations in kinematic and dynamic parameters. Direct transitions between controllers are demonstrated as well as parameterized control of changes in direction and speed. Feedback-error learning is applied to learn predictive torque models, which allows for the low-gain control that typifies many natural motions as well as producing smoother simulated motion. Figure 1.6 shows the input and output of the system. This work is published as [Yin et al. 2007]. The corresponding animation video can be downloaded at http://www.cs.ubc.ca/∼kkyin/animation/Yin_SIG07.mov.

Finally, Chapter 8 presents our conclusions, a discussion of limitations, and suggestions for future work.

Chapter 2

Related Work

In this chapter, we discuss the most relevant research to our work. The related work is divided into three areas: kinematic animation and optimization; dynamic simulation and robotics; and motor control. We begin by looking at related work in kinematic animation and optimization.

2.1 Kinematic Animation and Optimization

Computer animation requires the generation of trajectories. These can be generated in a variety of ways. For the purposes of this thesis, we classify techniques that use forward dynamic simulation as dynamic animation, and all other techniques as kinematic. Trajectory optimization techniques that incorporate physics-based equations of motion are considered as a kinematic technique subject to dynamic constraints.

2.1.1 Motion Editing and Data-driven Animation

Motion Capture [Bodenheimer et al. 1997], rooted in biomechanics (Section 2.3.1), has rapidly found its way in animation applications (films and games) and animation research during the last 10-15 years. It is currently mainly used as an animation acquisition technique (Figure 2.1(a)). One of the principle research challenges is to generalize motion capture data.

Motion editing in its simplest form can be modelled from a purely kinematic, signal processing point of view [Bruderlin and Williams 1995; Witkin and Popovic 1995; Unuma et al. 1995; Lee and Shin 1999]. Various signal processing techniques, including multiresolution filtering, displacement mapping, interpolation, extrapolation, warping and blending are applied to kinematic motion data.

Representative works that reuse motion capture data for motion synthesis and interactive avatar control include [Lee et al. 2002; Li et al. 2002; Kovar et al. 2002a; Arikan and Forsyth 2003; Arikan et al. 2003; Chai and Hodgins 2005]. Most of them use "resequence and blend" techniques. A statistically based approach is used

(a) (b)

Figure 2.1: (a) Motion capture in computer animation. (b) Figure 2.4 of [Patton 1998], courtesy of J. L. Patton. A sophisticated balance control experiment in Biomechanics.

to organize and index the motion database in [Chai and Hodgins 2005]. Machine learning, especially non-linear dimensionality reduction techniques, such as [Ghahramani and Hinton 1996; Tenenbaum et al. 2000; Roweis and Saul 2000; Hastie et al. 2001; Roweis et al. 2002; Matusik et al. 2003; Hertzmann 2003; Lee 2003; Teh and Roweis 2003; Salesin 2003] are finding promising applications in computer animation. [Lawrence 2004] uses MoG (Mixtures-of-Gaussian) model. Gaussian Processes have been successfully utilized for inverse kinematics in [Grochow et al. 2004].

Although data-driven animation systems are abundant, there is little work on modelling interactions with human motion. Notable exceptions are [Arikan et al. 2005; Komura et al. 2005a]). That is, areas marked by *A2*, Chapter 4 and *A3* in Table 1.1 are not well explored. This is likely because of the innate dynamic nature of push interactions.

2.1.2 Trajectory Optimization

The foundation of computer animation is the interpolation of key frames [Parent 2002]. Users start by specifing the position or orientation of some keyframes, which are represented by circles in Figure 2.2. Computers then generate in-between frames marked by triangles, resulting in a continuous trajectory. Usually splines of some form are chosen because of their smoothness. In the case of the trajectory in Figure 2.2, it

is continuous but not smooth because its derivative curve is not continuous. However, smoothness is just the simplest criterion for animation. There are cases we have more requirements. For example, the trajectories interpolated by splines may not be valid if we want a physically realistic animation. Imagine that the trajectory in Figure 2.2 specifies the trajectory of a ball. First it is thrown upwards, then it hits the ground and bounces. In the real world, the trajectory of a ball dropping and bouncing obeys Newton's laws. That is, the acceleration of the ball, computed by taken the second derivatives of the positions, has to be a constant for this case. The keyframes given by the users may not obey the laws of physics. Often we not only want a valid trajectory, but also the best trajectory given some goals for a motion. An optimization algorithm can be used to adjust the trajectory so that it minimizes an objective function while satisfying constraints. Possible objectives are energy cost, height reached, or distance traveled. Possible constraints are friction constraints, torque limit constraints, or ZMP(Zero Moment Point) constraints.

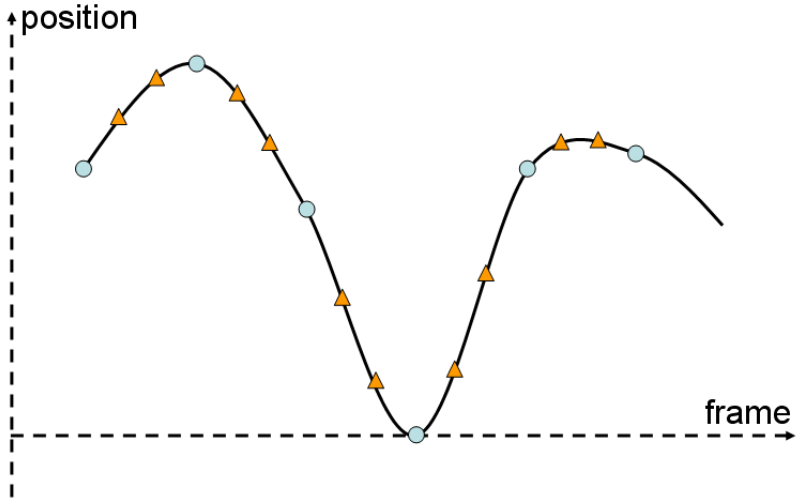


Figure 2.2: Trajectory interpolation and optimization. Circles are keyframes of the positions or orientations specified by the user. Triangles represent computer generated in-betweens.

Spacetime constraints (SC), first proposed by Witkin and Kass [1988], casts the motion editing problem into a constrained optimization framework. SC combines spatial kinematic constraints with temporal dynamics constraints. Conceptually, a constrained optimizer is used as the motor controller. This optimization approach has also been used in relevant neuroscience and movement science disciplines. These methods are usually computationally expensive and do not typically account for styles of motion. There are various improvements and extensions to the basic SC approach: [Cohen 1992; Ngo and Marks 1993; Liu et al. 1994; Rose et al. 1996; Gleicher 1998; Popovic and Witkin 1999]. They either try to make SC practical by addressing the inherent high cost of the optimization, or extend SC to address specific types of

animation tasks. Muscle dynamics can also be incorporated into the SC framework as in [Komura and Shinagawa 1997; Komura et al. 2001].

Spacetime optimization methods emphasize the physical plausibility of motions, by incorporating dynamic constraints into the optimization procedure. Dynamic constraints include satisfying the equations of motion, torque smoothness, torque limits, and foot-ground contact constraints. Foot-ground contact constraints often enter into the optimization as friction constraints or ZMP (Zero Moment Point) constraints. The ZMP concept is important to robot balance control, and will be discussed in more detail in Section 2.2.1. Here we briefly summarize the ZMP as the point where the horizontal moment of external forces and inertial forces balance. The larger the horizontal moment is, the further away the ZMP is from the center of foot. Because all physical systems have a finite foot length, the extra horizontal moment will simply make the system rotate and fall when the ZMP hits the foot boundary. The ZMP is mathematically equivalent to CoP (Center of Pressure).

[Fang and Pollard 2003] adopt physical constraints that have linear time analytical first derivatives (including the CoP constraint) to speedup the optimization of key-framed motions. [Liu et al. 2005] introduces nonlinear inverse optimization for estimating model parameters from motion capture data. Their dynamical model incorporates several factors of locomotion that are based loosely on the biomechanical literature. When used in a spacetime optimization framework, the parameters of this model define different styles of natural human movement. Motions can also be generated in the same style but performing different tasks, and styles may be edited to change the physical properties of the body. As with other spacetime results, the results are not directly shown to be capable of driving a forward dynamics simulation.

[Tak et al. 2000; Shin et al. 2003] are representative works that utilize the ZMP concept to have dynamic balance effects. They both concentrate on post-processing (touch-up or filtering) motion clips that might be physically implausible. For a robot or a physical system, the ZMP will never be outside of the support polygon, even when it falls. However, motions choreographed by animators manually might be violating physical laws, and the ZMP might be outside of the support polygon. The authors used an optimization framework to find the motion that is close to the original motion, yet the ZMP always resides inside or on the edge of the foot boundary. Realism of the motions can thus be enhanced by eliminating physically implausible part of an input motion. They are off-line motion editing techniques.

[Kudoh et al. 2002; Kudoh et al. 2006] are online balance controllers that can cope with pushes mainly in the sagittal plane. Appropriate sub-controllers can be selected based on whether they can recover from a particular perturbation. PD control is adopted for small perturbations and final posture adjustment, which is similar to our own work in Chapter 5. A quadratic programming controlling the ZMP constraint is activated when the perturbation is large, i.e., the ZMP is close to the boundary and the balance is in danger. The author claims real-time rates and realistic balance

reactions such as arm rotations. Stepping is activated when in-place control fails, similar to our work in Chapter 6. The main difference between their work and our work is that we construct controllers to drive dynamic simulations, while theirs use optimizations to generate trajectories.

The are several problems with the nonlinear optimization framework. It is unknown how to best formulate the objective and constraints to achieve given classes of motion. The computational cost is high, making it an offline tool. The optimization can easily get trapped in local minima.

2.1.3 Animation Interfaces

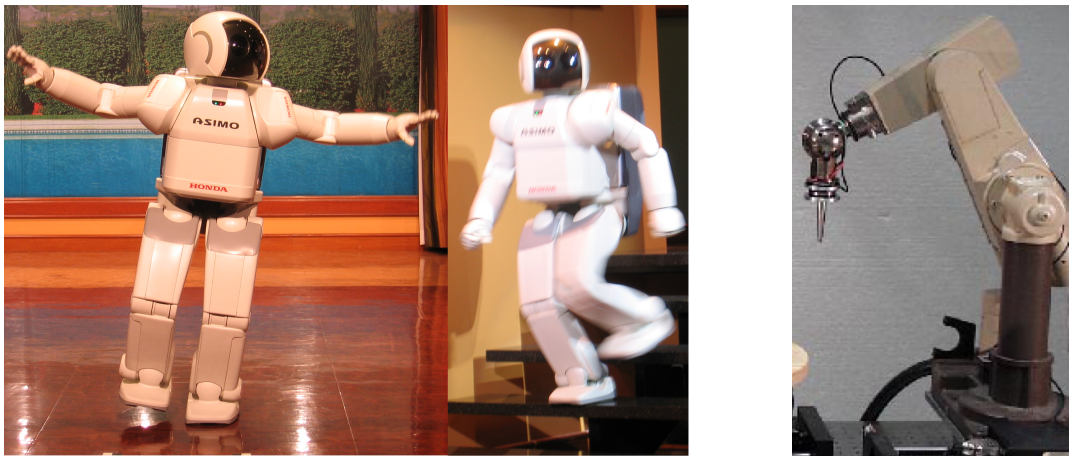
Character animation is difficult to specify and control, both for kinematic and dynamic systems. Kinematic approaches do not lead themselves to an obvious way as to how to choreograph with captured motion trajectories. Intuitive user interfaces play an important role in alleviating this problem. In recent years, sketch-based interfaces [Thorne et al. 2004; Yang et al. 2005], voice-driven interfaces [Wang and van de Panne 2006], and performance-driven interfaces [Lee et al. 2002; Chai and Hodgins 2005] have been very successful. Our work in Chapter 3 includes the use of foot pressure pads as a novel interface component.

2.2 Dynamic Simulation and Robotics

In this section, we focus on discussing previous work from computer animation and robotics literature that utilizes forward dynamic simulations or controls real robots.

Although robots have been successfully applied to "assembly lines" for automation in factories for years, bipedal walking robots are far less successful and robust. A fundamental reason why biped locomotion is hard is that the foot ground contact "joint" is unilateral and underactuated. Unilateral refers to the fact that the ground can only push but not pull the foot. Underactuated means there is no actuators between the ground and the foot that we can directly control. More intuitively, they mean the robots in Figure 2.3(a) do not have a fixed base as does the robot arm shown in Figure 2.3(b). As a result, legged robots have postural instability and can fall.

Recent years have seen considerable progress in humanoid robot research. Large corporations such as Honda [Hirai et al. 1998] and Sony have made amazing entertainment humanoids and human-sized experimental bipeds. They can walk [Yokoi et al. 2002] with different speeds to different directions, climb up and down stairs, kick, jog and even fan-dance. Other researchers are making robots that can lie down and get up [Hirukawa et al. 2003], and roll and rise [Kunivoshi et al. 2003]. However, great man power is involved to hand-pick the tasks, reconstruct and tune the controllers



(a) (b)

Figure 2.3: (a) State-of-Art Humanoid Robot: Honda ASIMO. (b) A typical industrial robot arm.

for the humanoids. Adaptive behaviour and learning are lacking. Robustness to large environmental perturbations is also unproven.

2.2.1 Balance Indices

The unique interaction characteristics between foot and ground are the fundamental reason that causes instability for bipeds. Most balance indices or postural stability criteria that incorporate dynamics are thus defined by the Ground Reaction Force (GRF). Only a conceptual understanding of the ZMP is sufficient for further reading of later chapters. Other indices are given for completeness, and some of them are relevant for future work. We avoid more formal definitions in the interest of understandability. Interested readers are referred to Appendix A for more details. We note that balance indices themselves do not directly describe what the controls should be in order to retain balance when performing tasks.

CoG and GCoM

We use GCoM (Ground projection of Center of Mass) and CoG (Center of Gravity) interchangeably in this thesis.

GRF and CoP

Suppose the quadrangles shown in Figure 2.4 are the feet. We define the CoP (Center of Pressure) as the point on the ground where the resultant normal ground reaction force acts. We decompose the interaction forces between the foot and the ground into:

- forces that are normal to the ground plane (Figure 2.4(a))
- forces that are tangential to the ground plane (Figure 2.4(b)).

The normal forces can only generate net torque vectors that are embedded in the horizontal plane (See Appendix A). The total resultant ground reaction force and torque, as shown in Figure 2.4(c), includes the normal force, the tangential force and the normal torque.

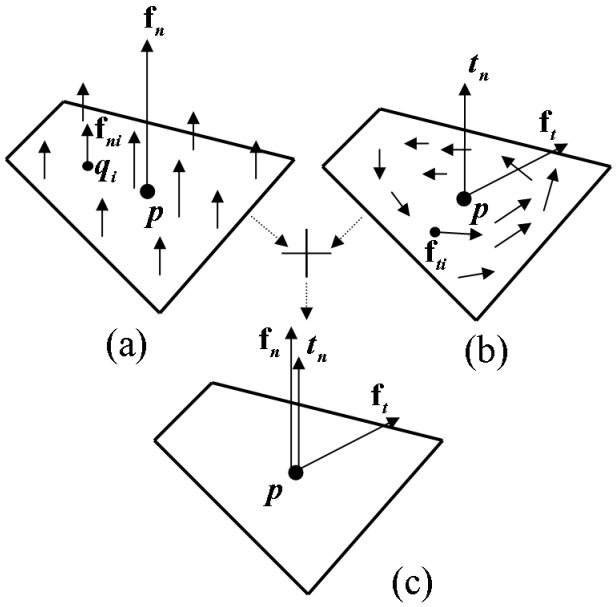


Figure 2.4: Illustration of the ground reaction force. $\mathbf{p}$ is the CoP. (a) Normal force and tangential torque. (b) Tangential force and normal torque. (c) Total ground reaction force and torque.

ZMP

The Zero Moment Point (ZMP), introduced by [Vukobratovic 1990] and widely used in the robotics literature, is defined as the point on the ground where the net moment of the inertial forces and gravity forces has no component along the horizontal axes. Mathematically the ZMP and the CoP (Center of Pressure) are equivalent [Goswami 1999]. The CoP is widely used in Biomechanics literature, while the ZMP is favored by robotics researchers. We refer the reader to Appendix A for the formal explanation of their equivalence.

The ZMP concept is important to robot balance control. ZMP is the point where the horizontal moment of external forces and inertial forces balance. The larger the horizontal moment is, the further away the ZMP is from the center of foot.

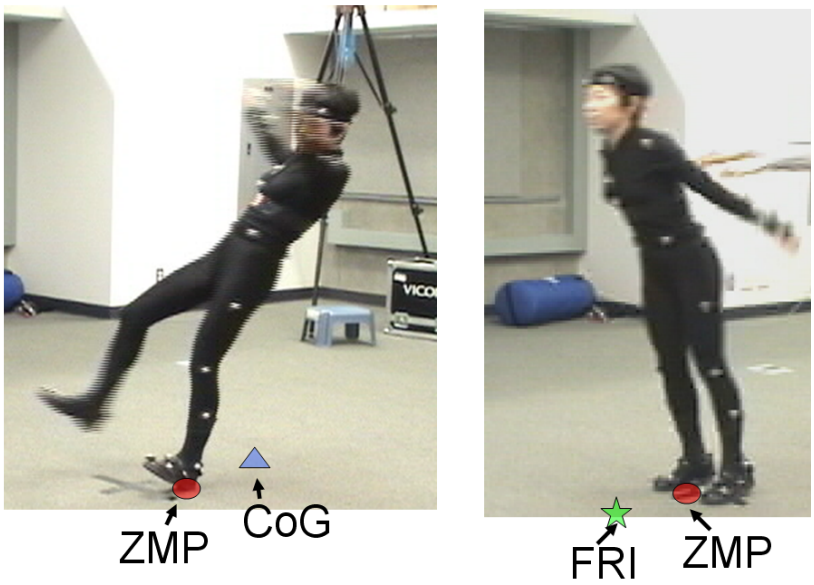


Figure 2.5: CoG, ZMP, and FRI

Because all physical systems have a foot of finite size, the extra horizontal moment will simply make the system rotate and fall when ZMP hits the foot boundary. Numerous robotics balance algorithms, such as [Ito and Kawasaki 2000; Kagami et al. 2000; Kim et al. 2002; Kajita et al. 2003a; Okumura et al. 2003; Ito et al. 2003; Zhu and Kawamura 2003], keep the ZMP inside the foot boundary all the time [1]. Keeping the ZMP inside the foot boundary is a necessary but not sufficient condition for balance. For a physically plausible motion, the ZMP is always inside or on the edge of the foot boundary, whether it is falling or not. See the left picture of Figure 2.5 for a comparison of the ZMP with the CoG.

The use of the ZMP as a postural stability indicator is limited in several ways. One is that keeping the ZMP inside foot boundary is not always possible. This is often not considered a problem for robotics because of the focus on algorithms that give maximum stability in benign environments. In computer animation, we wish to have balance algorithms that produce realistic motion in a wide variety of conditions. The Foot Rotation Indicator (FRI), introduced by [Goswami 1999] yet not widely used in robotics, is an indication of postural instability that is more general because it allows for foot rotation.

The conventional ZMP can be further generalized to the GZMP (Generalized Zero Moment Point) [Harada et al. 2003] to include hand-environment contacts, in the context of arm/leg coordination tasks.

[1] More formally, this can be referred to as the support boundary or base of support, which is the convex polygon of all the ground contact points. However, we will also refer to it as the foot boundary for simplicity and intuitiveness.

FRI

The significance of the FRI to our problem is that when in-place balance fails, the FRI can provide a hint of where to make a step to compensate for the unbalanced moment on the stance foot. We have not directly used this index in our stepping strategies, although this might be interesting to investigate further.

The FRI point is a point on the ground at which the resultant moment of the force/torque impressed on the foot is normal to the ground surface [Goswami 1999]. The impressed force/torque means the force and torque at the ankle joint, other external forces on the foot, plus the weight of the foot, but not the GRF. FRI and ZMP are closely related. Unlike the ZMP, the FRI can be outside of the support polygon, as shown in Figure 2.5. It is where the net ground reaction force would have to act to keep the foot stationary. Intuitively, we can think of the FRI as the virtual ZMP when the foot is infinitely large and static.

Momentum Control

A problem with the FRI is that it only serves as an instability measure for the single support phase of a biped with flat feet. Linear and angular momentum, on the other hand, is a more general quantity for motion planning and balance control of legged robots. One example is the Resolved Momentum Control in [Kajita et al. 2003b]. Various models and indices based on momentum control have been proposed, including IPM (Inverted Pendulum Model) [Kajita et al. 2001b], 3DLIPM (3D Linear Inverted Pendulum Model) [Kajita et al. 2001a], enhanced IPM [Kudoh and Komura 2003], ZRAM (Zero Rate of change of Angular Momentum) point [Goswami and Kallem 2004], and RMP (Reaction Mass Pendulum) model [Lee and Goswami 2007]. Momentum regulation is observed in human walking [Popovic et al. 2004]. A number of computer animation researchers are investigating momentum based editing techniques, such as [Abe et al. 2004; Komura et al. 2005a]. Our work in Chapter 4 uses momentum perturbations as motion index.

2.2.2 Equations of Motion and Control

There are two common ways of formulating Equations of Motion for an articulated body system: Newton-Euler and Euler-Lagrange [Featherstone 1987; Craig 1989; Schilling 1990; Murray et al. 1994; Mason 2001; Xie 2003]. We use a Lagrange-multiplier velocity-based Newton-Euler formulation. We describe our dynamics engine in more detail in Chapter 5. We use Open Dynamics Engine [ODE] for the work in Chapter 6 and 7.

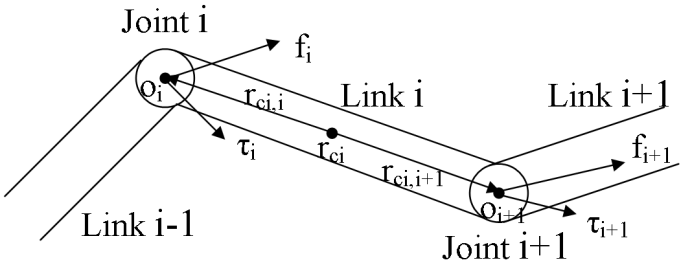


Figure 2.6: Illustration of the kinematics and dynamics of an articulated rigid body system.

Newton-Euler

As shown in Figure 2.6, for an articulated rigid body system, we use $\mathbf{f}_i, \tau_i$, to represent the force and torque applied to link i from link $i-1$ through joint i. $\mathbf{r}_{ci,i}$ is the vector connecting the center of gravity $\mathbf{r}_{ci}$ of link i to joint i's origin $\mathbf{o}_i$. $\mathbf{a}_i$ represents the linear acceleration at the CoM of link i . α_i is its angular acceleration. I_i is the inertia tensors of link i. All quantities are given in world coordinates.

The Newton-Euler formulation is recursive in nature. The backward Newton-Euler recursion relation is as follows:

$$\mathbf{f}_i = \mathbf{f}_{i+1} + m_i\mathbf{a}_i - m_i\mathbf{g} \tag{2.1a}$$

$$\tau_i = \tau_{i+1} - \mathbf{r}_{ci,i} \times \mathbf{f}_i + \mathbf{r}_{ci,i+1} \times \mathbf{f}_{i+1} + \mathsf{I}_i\alpha_i + \omega_i \times (\mathsf{I}_i\omega_i) \tag{2.1b}$$

Euler-Lagrange

The Euler-Lagrange formulation has a compact description of motion equations in matrix form. The equations of motion can be written as:

$$\mathsf{M}(\theta)\ddot{\theta} + \mathsf{C}(\theta, \dot{\theta})\dot{\theta} + \mathbf{g}(\theta) = \tau \tag{2.2}$$

where θ represents the generalized coordinates and τ is the generalized forces. When we adopt joint angles as θ, τ represents the joint torques. Matrix $\mathsf{M}(\theta)$ is the inertia matrix. Matrix $\mathsf{C}(\theta, \dot{\theta})$ accounts for centrifugal and Coriolis effects[2]. Vector $\mathbf{g}(\theta)$ accounts for the effect caused by gravitational force.

Null Space Postural Control

Equations of motion govern how the system evolves over time given the controls, taking the form of joint torques τ in our case. When there is only a single task to

[2]In the classical mechanics, one identifies terms of the form $\dot{\theta}_i\dot{\theta}_j, i \neq j$ as Coriolis forces and terms of the form $\dot{\theta}_i^2$ as centrifugal forces.

perform, we may attempt to solve the control problem by direct formulation and manipulation of motion equations. However, a robot typically has to maintain its posture or balance while accomplishing a specified manipulation task. Researchers have developed algorithms, such as [Park et al. 1996; Crowe et al. 1998; Chang and Khatib 2000; Kim et al. 2002; Khatib et al. 2004], based on the idea of kinematic or dynamic decoupling of the control associated with different tasks. The most important task is achieved by solving the motions equations in the full control space, torques for secondary tasks are only performed in the null space of the first task space. This formulation allows for posture objectives to be controlled without dynamically interfering with the operational task.

2.2.3 Learning from Humans

Robotics researchers often have the robustness of their control algorithms as a primary concern. An alternative objective is to investigate how to make robot motions more human-like [DasGupta et al. 1998; Safonova et al. 2003; Kagami et al. 2003], targeting applications such as entertainment robotics or demonstrations of motor learning.

A desired ZMP location only introduces a 2-dimensional constraint. Common ZMP algorithms choose to modify one major joint, such as the waist joint [Nakaoka et al. 2003], to meet the ZMP requirement. In contrast, humans are much better in multi-joint coordination. Motion capture technology, as mentioned in Section 2.1.1, provides a starting point to learn motor skills from humans.

Current control algorithms, including sophisticated ones with many parameters to tweak, lack the ability to self-evolve and adapt, and rely on researchers to carefully design every behavior. Humans have the ability to balance motion goals with kinematic and dynamic constraints, and to acquire new motor skills and adapt old ones. Additional insights into human motion will be discussed in Section 2.3.2.

2.2.4 Dynamic Simulation for Animation

Dynamic simulation for animation has been demonstrated in many different scenarios and at many different levels of detail. Neuromusculoskeletal simulations focus on more detailed modelling of human motion, such as modelling the neck [Lee and Terzopoulos 2006] or limbs [Pai et al. 2005]. For full-body motion simulations, muscles are often abstracted as joint torques. Air-borne stunts [Wooten 1998; Zhao 2004] are more successful than motions that involve environmental interactions such as locomotion tasks [Hodgins et al. 1995; Faloutsos et al. 2001], in terms of perceived motion realism. Control of balance remains an open problem. In the game industry, rag doll simulations are common but avoid issues of balance. Recent years have seen more pseudo-physics to deal with the problem [Wrotek et al. 2006], by using a virtual helping force or the "hand of God". Hybrid kinematic and dynamic systems have also

been proposed, such as [Zordan et al. 2005], where the initial impact phase is simulated, but balance recovery is achieved by blending with kinematic motion capture examples.

Balance control in simulated computer animation basically can be classified into static balance control and dynamic balance control. Static balance requires the GCoM always remain within the support polygon. For a stationary system, GCoM, ZMP, FRI all coincide. Algorithms controlling GCoM can achieve good results for static and slowly moving systems. GCoM is also easy to understand. Thus it has been the most commonly used balance control variable by computer graphics researchers [Zordan and Hodgins 1999; Zordan and Hodgins 2002; Zhao 2004; Zhao and van de Panne 2005].

Highly robust dynamic balance control for stepping and locomotion, implemented using forward dynamic simulation or on a robot, has yet to be seen. Our work in Chapters 6 and 7 makes significant progress in this direction by using a change-of-support strategy when the CoM falls out of the support polygon. Meanwhile, dynamic balance considerations are sometimes incorporated into off-line postprocessing or interactive motion synthesis in kinematic modelling, as discussed in Section 2.1.2 and our work in Chapter 4.

2.3 Motor Control

While some human motions have been successfully reproduced on humanoid robots, the quality of many robot movements can be characterized as slow, rigid, and clumsy. While on occasion such robotic movements may be useful in computer animation, the more common desire is to replicate realistic motion. The simulated or synthesized motions have to be of high quality in order to be seamlessly blended with realistic motions prepared by motion capture. How can we make robotic motions more "human"? Motor control in biomechanics and neuroscience may provide some insights. Research in these fields is far from being able to fully understand the generation of human movement. Nevertheless, computer animation can benefit from taking advantage of principles and models that have been the product of research in these disciplines.

2.3.1 Biomechanics

Biomechanics is the science that examines forces acting upon and within a biological structure, and effects produced by such forces [Hogan 1990; Alexander 1992; Adrian and Cooper 1989; Nigg and Herzog 1995; Hall 1998]. We can think of biomechanics as "Mechanics on living things", in contrast to "mechanics on electromechanical components". Human biomechanics is relevant to our problem in at least three ways:

- In a typical human motion simulation, the segment inertia properties have to be known.
- If a PD control law is to be used for human simulation, the joint stiffness and damping coefficients have to be known.
- If more complex control laws that include muscle dynamics are used, the muscle dynamics properties have to be known. Many muscle models have been proposed over the years. The Hill model [Winters and Crago 2000; Hogan 1990] is commonly used because of its simplicity.

Motion capture for biomechanical analysis of motions is often more involved than motion capture used for computer animation. Figures 2.1(a) and 2.1(b) illustrate this. A biomechanics experiment often requires measuring devices that can measure

- External forces and moments. For example, computer controlled robot devices are used for delivering fixed perturbation/interaction forces at known times; force plates are used for measuring GRF.
- Internal muscle activations. An example would be surface electromyographic (EMG) electrodes.
- Kinematic quantities. The procedures are roughly the same as motion capture in computer animation, but with stricter requirements on the measurement accuracy.

Our measurement of foot-ground pressure in Chapter 3 is inspired by GRF measuring from force plates in biomechanics.

2.3.2 Computational Motor Control

As we see in Section 2.2.3, motion capture only provides a starting point for learning to produce motions based upon human motion data. Eventually we wish to develop a motor control model from the observed motions. Neuromotor control is a branch of neural science that seeks to explain biological motor behavior at a low level [Kandel et al. 2000; Winters and Crago 2000; Wise and Shadmehr 2002]. The computational study of biological motor control is often called Computational Motor Control [Mussa-Ivaldi 1999; Jordan and Wolpert 1999; Wolpert and Ghahramani 2000]. It is fundamentally concerned with the relationship between sensory signals and motor commands. There are two basic transformations involved. The first transformation is from sensory signals to motor commands. This is often referred to as motion planning. The second transformation is from motor commands to their sensory consequences and is often referred to as motion execution. It is governed by the physics of the environment, the musculoskeletal system and the sensory receptors. The human motor

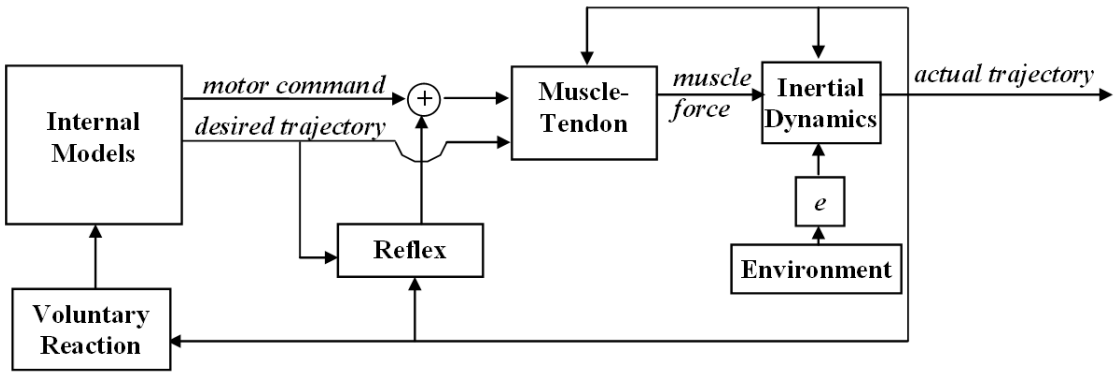


Figure 2.7: A complete motor system.

system consists of peripheral and central parts, located in the peripheral and central nervous system respectively.

Motor systems are often decomposed into the feedforward components and feedback components [Jordan and Wolpert 1999; Kandel et al. 2000; Wang et al. 2001]. Figure 2.7 shows a typical motor control model. Internal models are thought to output a feedforward motor command and desired trajectory. The muscle-tendon system is driven by the motor command and generates muscle forces. Muscle forces and external forces act upon the human inertial dynamics system and produce the actual trajectory.

There are three feedback mechanisms: muscle-tendon feedback which has essentially zero delay because it involves the passive properties of muscles and tendons; reflexes, having >30 ms delay; and voluntary reactions involving cortical replanning which have a delay of >100 ms. The feedback loops with longer delays are higher in the control hierarchy of the motor system, and are thus less well understood and more challenging to model.

In computer animation and robotics, research has already examined the motor control problem to some extent [van de Panne and Fiume 1993; Sims 1994; Grzeszczuk and Terzopoulos 1995; Grzeszczuk et al. 1998; Smith 1998; Hornby et al. 1999; Tedrake and Seung 2002], mainly focused on parametric search methods and neural networks techniques. However, a systematic exploration of neuromotor control and computational motor control reveals that there is more that can be adopted. We note that neuromotor control is a broad field with diverse hypotheses. Thus, we only review the specific principles most related to our problem here.

Two Views of Motor Control

What is the motor system controlling during a motor task? There are numerous hypotheses, classifiable into two categories:

- Control objectives are kinematic quantities, such as position, velocity and ac-

celeration, and the variance of end-effector location. A well known kinematic model is the minimum jerk model [Flash and Hogan 1985]. Smoothness and invariance of the motion trajectories are deemed to be the fundamental characteristics of human movements. There are also many empirical observations such as motor primitives, two-thirds power law, Fitts' law, and Donders' law.

- Control objectives are dynamic quantities, such as energy consumption, muscle stiffness/impedance/compliance, muscle torques, or interaction forces. Minimum torque-change is a commonly used dynamic model [Uno et al. 1989].

Both kinematics and dynamics likely provide control objectives for the motor system. Depending on the task, goal and environment, one may be more important than the other. In an optimization framework, the important one should probably be in an objective function, while the less important one can serve constraints. This stems in part from work in neuroscience and biomechanics [Wise and Shadmehr 2002] that the motor system has both kinematic and dynamic representation of movements. For example, the Posterior Parietal Cortex (PPC) and Premotor cortex (PM) are believed to have key roles in formulating a kinematic plan; Primary Motor cortex (M1) and cerebellum are known to be more related to the dynamic implementation of a motor plan. In practice, computational neuroscience, robotics, and computer animation have used both kinematic and dynamic objective functions in different situations in various forms. Under most circumstances kinematic planning is likely higher in the motor control functional hierarchy, and happens earlier in the motor control process [Rosenbaum et al. 1999; Kandel et al. 2000; Wise and Shadmehr 2002].

Internal Models

Internal models (IM) represent one of the most successful concepts established in neuroscience in recent years [Kawato 1999; Wise and Shadmehr 2002]. Internal models are neural mechanisms that can mimic the input/output characteristics, or their inverses, of the motor apparatus. During a typical motor task, the brain may solve three types of computational problems, as illustrated in Figure 2.8:

1. Forming a kinematic plan. This requires learning a model of the forward and inverse kinematics of the controlled system and its environment in order to know how to accomplish the goals of the task. However, since there are usually redundant DoFs in biological systems, there is no unique kinematic solution to the planning. The internal kinematics models should provide the ability to choose the best solution. There are many controversial hypotheses in terms of what the brain is controlling or optimizing during motor planning.

2. Inverse dynamics. This requires learning a model of the inverse dynamics of the controlled system. This model is usually called the inverse internal model.

It answers the following question: how can the body be made to move along a planned trajectory? More formally, it provides the ability to predict the inputs that should be provided to the biomechanical system for a given desired change in state of that system. Inverse internal models can calculate necessary feedforward motor commands from desired trajectory information.

3. Forward dynamics. This requires learning a model of the forward dynamics of the mechanical system. This model is usually called the forward internal model. It answers the following question: what will happen to my limbs and/or the environment if I activate my muscles in a certain way? More formally, it provides the ability to predict how the biomechanical system will behave as a function of current input. Forward internal models can predict sensory consequences from efferent copies of issued motor commands, which is useful in the context of systems that have inherent delay.

Internal models are a blanket term used to describe the information contained in the solution to these three types of computational problems, and motor memory refers to the representation of IMs in the Center Nervous System (CNS).

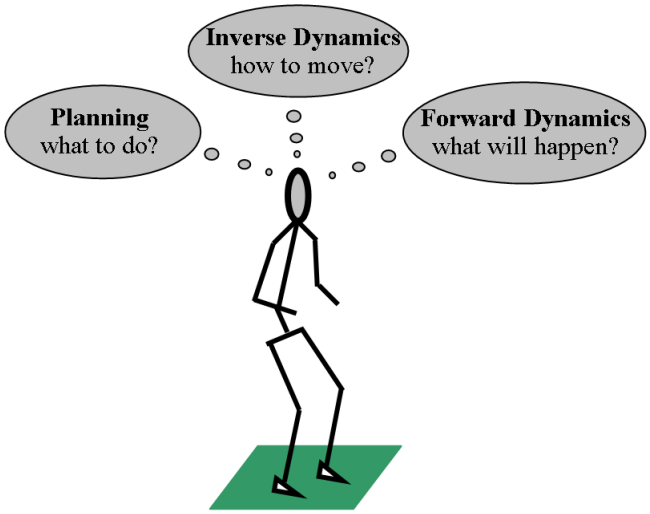


Figure 2.8: Internal Models

The necessity for IMs in motor control originates from the fact that biological motor systems, including neurons and muscles, are very slow. The fastest reflex response to the sensory information about the perturbation requires about 50 ms to travel from the sensors to the spinal cord and out along the motor axons, followed by another 50 ms delay for the processes involved in excitation-contraction coupling to change the force output of the muscle. Very fast voluntary movements take about 250 ms. The time delay of 100 ms occupies a large proportion of movement execution time of 250 ms, and therefore the feedback gains cannot be set high to avoid instability. In

contrast, robots can work at high (0.5-10 kHz) sampling and control frequencies which supports the implementation of high feedback gains. Fast and coordinated movements cannot be executed solely under feedback control in biological motor systems, since the biological feedback loops are slow, and have small gains. Thus, the internal model hypothesis proposes that the brain needs to acquire an inverse dynamics model of the object to be controlled through motor learning, after which motor control can be executed in a pure feedforward manner. In theory, a forward model of the motor apparatus embedded in an internal feedback loop can approximate an inverse model.

The significance of the IM concept to computational motor control is that the existence of IM shows that biological motor control fundamentally differs from robot control. Robots can use high-gain fast feedback loops to do trajectory tracking, while humans are thought to rely on IMs to compensate for slow feedback loops. Although we can borrow robot control mechanisms in simulation for animation purposes, the resulting motions and perturbation responses are not going to look human unless extra tricks are incorporated, such as the gain scheduling procedure in [Zordan and Hodgins 2002]. In Chapters 5 and 7 we use feedforward components based on the IM concept to achieve more stable biological motor control and more natural perturbation responses, even though the mechanisms of IM acquisition, representation, coordination, and execution is not exactly known today.

Synergy

A fundamental problem of motor control is what is known as the *degrees of freedom* problem. As an illustrative example, suppose a person raises her hand to adjust her glasses. There are twenty-six muscles – ten at the shoulder, six at the elbow, four at the forearm, and six at the wrist – that govern the movement of the four arm joints we are moving. Even simple movement of the arm requires the complex temporal and spatial orchestration of all these muscles. Each joint must move the correct amount and at the right time in order for us to reach our destination and accomplish our task. The problem of managing this complexity is known as the degrees of freedom problem. The human body, considered as a movement system, has an astronomical number of states with the hundreds of thousands of muscle fibers that make up the six hundred and ten muscles that regulate the motion at the joints formed by our two hundred and ten bones [Goldfarb 1993].

How can we handle all these degrees of freedom? How do we learn new motor tasks? The computations are daunting in complexity if the potential degrees of freedom are not temporarily reduced. This reduction of DoFs is accomplished by adding constraints. A *muscle synergy* constrains the relative activity of muscle groups that span two or more joints, thereby linking them together. The neural basis for muscle synergy is that in the descending pathway of the nervous system, one cortical neuron often projects through synapses to several inter-neurons, and one inter-neuron often

projects to several motor neurons. Thus, cortical output neurons or spinal inter neurons do not map one-to-one to muscles. Although muscle synergy is a concept related to dynamics, synergy itself has a much broader meaning. Synergy can be either kinematic or dynamic, spatial or temporal or spatio-temporal, associated with neurons, muscles or limbs. Synergy means the movements of joints are often correlated in a certain fashion. Humans are capable of moving their arms in many peculiar ways, such as moving one segment at a time when trying to imitate a robot. However in every day life, we tend to move our arms in a highly stereotyped and stylized way with stable spatial and temporal characteristics.

Although synergy appears to be an attractive concept, much of current work in this area is based on either PCA (Principal Component Analysis) or ICA (Independent Component Analysis). [Santello et al. 1998] found that the first two principal components could account for >80% of the variance of the 15 digit joint angles, in grasping a large number of familiar objects. The first principal component corresponds roughly to how wide the object to be grasped is, and the second principal component corresponds roughly to how long the object is. This hints that by parametrizing the object shape, we can actually get the principal components, which we can then use to synthesize motion. A problem with PCA is that when kinematic constraints are present, even with the principal components accounting for 99% of the variance, the resulting motion will still violate the constraint. As observers, humans are known to be sensitive to errors in constrained motions, such as the footskate problem when a foot ground contact constraint is violated [Kovar et al. 2002b]. In contrast, motion such as a hand waving goodbye tolerates a much higher error in terms of how much the resynthesized motion can differ from a known ground truth motion while still being plausible.

Postural Reflexes

Reflexes are the middle level feedback mechanism, slower than the muscle-tendon feedback, but faster than cortical feedback. After a perturbation most postural responses are reflex mechanisms, especially the initial responses. We will first discuss reflexes in general, and then look at postural reflexes in particular.

Reflexes are stereotyped, adaptable motor behaviors evoked by specific sensory stimuli [Kandel et al. 2000]. Although they can involve both spinal and supraspinal pathways, they are generally fast and functionally limited. Voluntary movements, however, require integration of more sensory information, such as visual, vestibular, and somatic sensory, and planning in cerebral cortex. The fastest voluntary reaction time is about 250 ms, while the slowest reflex takes less than 200 ms. Thus reflexes are important in providing rapid responses to perturbations or disturbances of movement. There are many kinds of reflexes, and we classify them into two categories:

- Spinal Reflexes: reflexes mediated by neural circuits entirely confined to the

spinal cord. They are generally very fast (<80 ms). They can further be classified to two types:

  - Monosynaptic: reflexes with sensory neurons and motor neurons directly connect to one another, with no interneuron intervening between them. Stretch reflexes, such as the well-known patellar reflex (also known as knee-jerk), are monosynaptic reflexes. This type of reflex has very short latency, and can be as fast as 30 ms.
  - Polysynaptic: reflexes including one or more sets of interneurons. Most reflexes, such as the withdraw reflex after touching a hot stove, are polysynaptic reflexes. Because more neurons and synapses are involved, this type of reflex has long latency, usually >45 ms.

- Long-loop Reflexes: reflex responses mediated via supraspinal structures. Both cortical and subcortical regions might be involved. Cortical routes mainly regulate distal muscles, which are responsible for tasks requiring precise regulation by voluntary commands. Subcortical routes mainly regulate proximal muscles, which are most responsible for more automatic motor functions, such as maintaining balance and producing gross bodily movements. Their time scale is usually between 80-160 ms. Long-loop reflexes allow responses to be modulated and adapted by supraspinal centers, sometimes even to the extent of reversing movements when appropriate. An example is the phase-dependent reflex reversal of the obstacle-induced tripping response during locomotion.

Postural reflexes are the collection of reflexes that regulate postural control, which are detailed in the following section. Researchers document reflexive postural adjustment behaviors of human subjects in many different balance tasks, and extract knowledge about balance control strategies. Postural reflexes are semiautomatic responses (70-180 ms) to a disturbance or perturbation, occurring after a monosynaptic response, and before a voluntary reaction can occur. They are subcortical long-loop reflexes and long latency spinal reflexes. Short latency stretch reflexes do provide the first defense line after a postural perturbation, but they are generally not considered to be postural reflexes. Due to the inherent difficulty of biped balance control and locomotion, more supraspinal influences are observed for balance control of humans than in, say, cats. These claims are supported by the literature which will be reviewed shortly in the following section. The exact mechanism of how supraspinal structures select and regulate reflexes to be task, functional and phase appropriate is largely unknown.

2.3.3 Postural and Balance Control Strategies

We have discussed the broad topic of motor control. We now concentrate on a literature review of the particular problem of balance control and postural adjustment. The mechanical problem of maintaining posture is particularly challenging for erect bipeds. However, humans can make proper postural adjustments usually without thinking, i.e., through subconscious control. From the discussion in the previous section, we see that they are mostly spinal and subcortical. The human neuromuscular postural system must meet at least three main challenges. It must maintain a steady stance in the presence of gravity; it must generate responses that anticipate volitional goal-directed movements, and recover from unexpected external disturbances; and it must be adaptive.

According to our knowledge, currently there is no general model available for postural and balance control. Researchers are still striving to discover and understand individual low level balance strategies or mechanisms. We summarize a list of such strategies here, and believe they can help define how a successful balance controller should behave. These previous works are enlightening in terms of experiment design, data acquisition, and data analysis.

We mainly focus on studies on standing, stepping and locomotion postural control, and balance recovery. All these balance mechanisms are interrelated physiologically. That is, one mechanism may be implicated in multiple balance tasks, and one task can utilize many mechanisms. We first summarize a list of balance strategies. Then a more detailed literature review is given. Findings from these studies support the major points in Section 2.3.2.

Summary of postural strategies

There are four characteristic postural strategies commonly observable kinematically in postural responses to external perturbations [Horak et al. 1997; Buchanan and Horak 2001; Shumway-Cook and Woollacott 2001]: the ankle strategy, hip strategy, arm strategy, and stepping strategy. We will utilize these strategies in Chapters 4 and 6. Instead of being "discrete" strategies, in humans they form a continuum of "mixed" strategies in postural responses. Similarly, in terms of control, neither a simple reflex mechanism nor a fixed muscle synergy organization is adequate to explain the muscle activation patterns observed in postural control tasks. A flexible continuum of muscle synergies that are modifiable in a task-dependent manner are used for postural control [Henry et al. 1998]. We summarize the four strategies as follows:

1. The *ankle strategy* uses distal to proximal muscle activation. This strategy is characterized by body sway resembling a single-segment-inverted pendulum. It is typically elicited during small shifts on flat support surfaces or perturbations

of CoM when the task requires maintenance of upright posture.

2. The *hip strategy* uses early proximal hip and trunk muscle activation. This strategy is characterized by body sway resembling a double-segment inverted pendulum divided at the hip. It is typically elicited during perturbations that are large in comparison with the supporting surface (short surface), or on compliant support surfaces, or when the task requires a large or rapid shift in CoM.

3. The *arm strategy* uses upper arm muscle activation. This strategy is characterized by rapid arm rotation. It is typically elicited during perturbations that are too large to recover from using just lower limb strategies, and with the presence of surface or instructional constraints not to step.

4. The *stepping strategy* uses early activation of hip abductors and ankle co-contraction. This strategy is characterized by asymmetrical loading and unloading of the legs to move the base of support under the falling CoM. It is typically elicited when there are no surface or instructional constraints, or when the perturbations are extremely large and in-place balance is not possible. Multiple steps may occur during balance recovery.

Postural control during standing

Various strategies have been reported for standing postural control. [Kuo and Zajac 1993; Rietdyk et al. 1999] report that both "hip strategy" and "ankle strategy" are present in controlling the CoM during postural response to perturbation in standing posture; and the "hip strategy" is more effective than "ankle strategy". [Runge et al. 1999] indicate that hip strategy is added to ankle strategy to produce a continuum of postural responses, by analyzing the strategies on the joint torque level. [Rietdyk et al. 1999; Winter et al. 2003] also indicate that CoP is the variable used to control the CoM. [Hoogvliet et al. 1997] propose that a "foot tilting strategy" is utilized to control the CoP during one-leg stance.

Standing postural control has been principally modelled as an inverted pendulum. [Pai and Patton 1997] used an inverted pendulum model to identify that the environmental (contact force), anatomical (foot geometry), and physiological (muscle strength) constraints all play a role in limiting balance recovery, which conforms to experimental findings. [Pai et al. 2000] further prove that dynamic models instead of static models should be used in determining the threshold where a compensatory step must be initiated in order to recover balance. [Winter et al. 2003] develop a 0th order inverted pendulum model, and a non-linear model consisting of a series of elastic elements representing the plantarflexor. [Peterka 2003] evolved a model with PD control and feedback loops including visual, vestibular, proprioceptive and force sensory information.

Balance control involving stepping

There has been much work on balance control which involves stepping. [Eng et al. 1997] identified intralimb dynamics that utilizes passive dynamics during quick corrective action caused by unexpected mechanical perturbations applied to the foot during early and late swing of walking. [Rietdyk et al. 1999] report that the first line of balance defense is provided by muscle stiffness, not reflex-activated muscle activity. [Rietdyk and Patla 1998] studied balance recovery from tripping in unilimb support and trilimb support. Their results show that reflexive responses are modulated according to the balance requirements to optimize the recovery strategy. [Maki et al. 2003] review their studies on change-in-support balance reactions, including compensatory stepping and grasping. These reactions are much more rapid than volitional limb movements.

[Hsiao and Robinovitch 1999] develop a pendulum-spring model for balance recovery by stepping, and predict that successful balance recovery by stepping is governed by a coupling between step length, step execution time, and leg strength. [Patla et al. 1999] require participants to avoid stepping on light spots under different time constraints during locomotion. They find that alternate foot placements are chosen systematically to minimize required changes to the ongoing locomotor muscle activity, and the posed threat to dynamic stability. [Maki and McIlroy 1999] find that for compensatory stepping reactions, single-step reactions favor stability over speed, whereas multiple-step reactions are distinctly different and favor speed.

Results from [McIlroy and Maki 1999] suggest that anticipatory control is not the primary mechanism by which the CNS deals with the lateral instability arising during rapid compensatory stepping reactions. Yet [Zettel et al. 2002] claim CNS may be acting to avoid potential risk by adjusting anticipatory postural adjustments (APAs) accordingly during triggered stepping reactions evoked by unpredictable perturbations. [Marigold and Patla 2001] study the effect of prior experience and knowledge in slippery surface locomotion. [Marigold et al. 2003] further investigate the role of the unperturbed limb and arms in the reactive recovery response to an unexpected slip. [Chow et al. 2002] examine the effect of sudden release load on CoP motion and the response strategy with respect to load. Their interesting finding is that humans are prepared for the worst in balance control.

Balance control during locomotion

[Patla 2003] provides a good survey of strategies for dynamic stability during adaptive human locomotion, including reactive control, predictive control and anticipatory control. [Hirschfeld and Forssberg 1991] study the treadmill walking and find the modulations of anticipatory postural activity was phase-dependent. [Schillings et al. 1999] investigate reflex responses of subjects walking on a treadmill while the forward sway of the foot was unexpectedly obstructed by an obstacle. All subjects show phase-

dependent short-latency stretch reflexes, which is the first line of defense in preparing for the functional reaction generated by longer latency responses. [Schillings et al. 2000] further the stumbling-over-obstacle study by identifying phase dependent long-latency reflexes as well. These functionally important response strategies depend on long latency responses and are assumed to be premotoneuronal in origin. Similarly [Eng et al. 1994] report functionally appropriate responses for recovering from obstacle induced trips in different stages of human walking.

Chapter 3

Data-Driven Kinematic Animation from Foot Pressure: FootSee

This chapter presents our first contribution, which develops a data-driven kinematic animation technique for interactive performance-based animation. Data-driven kinematic animation has become a popular approach in recent years. It is able to generate high quality motions without having to fully tackle the motion control problem.

Our system, named FootSee, is an intuitive animation interface that uses a foot pressure sensor pad to interactively control avatars for video games, virtual reality, and low-cost performance-driven animation. During an offline training phase, we capture full body motions with a motion capture system, as well as the corresponding foot-ground pressure distributions with a pressure sensor pad, into a database. At run time, the user acts out the animation desired on the pressure sensor pad. The system then tries to "see" the motion only through the foot-ground interactions measured, and the most appropriate motions from the database are selected, and blended online to drive the avatar. FootSee can control a virtual avatar in a fixed latency of one second with reasonable accuracy. Thus makes it possible to create interactive animations without the cost or inconveniences of a full body motion capture system. An example result is shown in Figure 3.1, where foot pressure data generates an animated kick.

We start by briefly describing the motivation for performance-driven animation systems, and give an overview of our system in Section 3.1. Section 3.2 discusses closely related work. We then give a detailed description of how to gather and process data (Section 3.3), recognize new motions to select good matches (Section 3.4), and edit selected motions to generate new motions (Section 3.5). Experimental results are given in Section 3.6. The final section in this chapter provides a discussion of limitations and possible directions for future work.

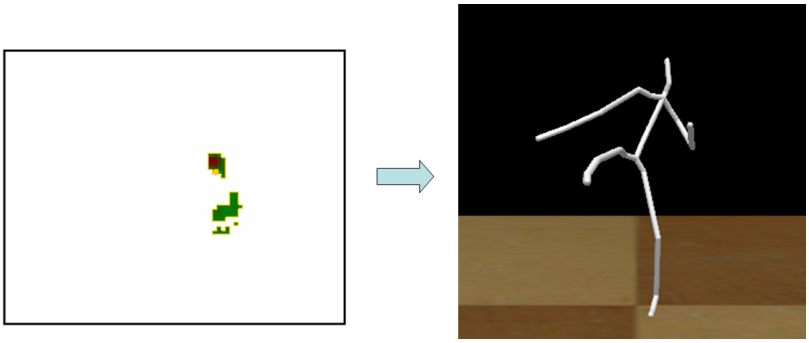


Figure 3.1: An example result. A kick motion is synthesized by the system to follow a user's foot pressure pattern.

3.1 Motivation

Avatar control in video games, virtual reality, and human-like character animation has been widely studied (see [Cavazza et al. 1998] for a survey). There are several reasons why this problem is challenging. Most animation packages have keyframe-based interfaces for creating new motions, and are only mastered by experts. More exotic interfaces, such as [Oore et al. 2002], are more intuitive, but not very efficient or general. To make matters worse, human observers are really good at human motion perception; some recent studies of human motion perception include [Hodgins et al. 1998; Reitsma and Pollard 2003].

Although dynamic simulation techniques have been used to animate humans [Hodgins et al. 1995; Faloutsos et al. 2001], data-driven kinematic animation systems that reuse motion capture data provide superior realism, larger motion repertoires, and richer styles. Figure 3.2 shows a typical decomposition of data-driven kinematic animation systems.

Data-driven kinematic animation systems consist of two major components: (1) a front-end user interface that enables users to direct and control desired new motions. (2) a back-end synthesis algorithm that uses the input commands to select and edit motions from the motion database. Previous work has demonstrated striking similarity in their back-end algorithms [Kovar et al. 2002a; Arikan and Forsyth 2003; Lee et al. 2002]. What makes these systems more interesting is the variations in the front-end user interface. Traditional keyboard and mouse based user interfaces cannot always meet the desired requirements of character animation systems. Voice-driven [Wang and van de Panne 2006], sketch-based [Thorne et al. 2004], and performance-driven interfaces [Lee et al. 2002] are proposed to alleviate this problem.

Motion capture based performance-driven character animation has become popular and widely used [Lee et al. 2002; Chua et al. 2003; Bodenheimer et al. 1997; Belland et al. 2002]. However, full body motion capture is expensive and delicate.

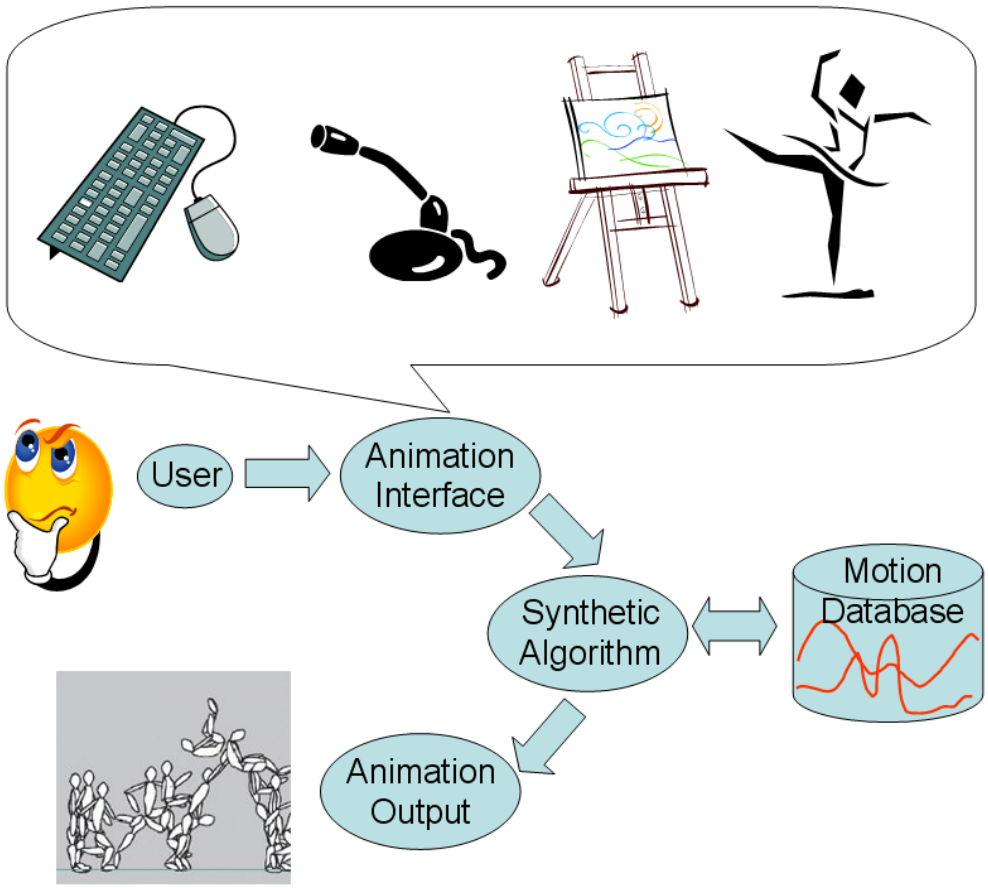


Figure 3.2: Data-driven kinematic animation system.

The capturing process involves significant effort including careful preparation, calibration, special clothing, and tedious post-processing. Less expensive and non-intrusive capture devices, such as the foot pad used in Dance Dance Revolution (http://www.ddrgame.com), the camera in Sony EyeToy (http://www.ddrgame.com), or the motion sensor in Nintendo Wii (http://wii.nintendo.com), have gained popularity in real-world applications. Despite their limited functionality, these type of "motion capture in the living room" interfaces have a great potential in the entertainment market.

Our system, named FootSee, uses a non-intrusive easy-to-use pressure sensor mat as the animation front end. A high quality motion database serves as the animation back end. Our hypothesis is that many full body motions have distinctive foot pressure signature. In addition, postural control is an essential part of all full body motions, and is constantly regulated by foot-ground interactions. Our work provides a viable alternative to applications that need an intuitive, interactive, robust, and high quality animation interface, but do not need high accuracy reconstruction (i.e., require only plausible animations and can afford occasional mistakes), and do not involve subtle

motions of the upper body.

There is rich information encoded in the foot-ground interaction forces. Force platforms (often called force plates) have been widely used to measure ground reaction forces (GRF) for clinical gait analysis, athletic performance analysis, rehabilitation, kinetic and biomechanics research [Tyler-Whittle 2002; Winters and Crago 2000; Alexander 1992; Nigg and Herzog 1995; **?**]. In these applications, one typically measures the total force and moment. In contrast, we measure the normal pressure distribution to extract information about locomotion and postural control. To our knowledge, there has been no application of the foot-ground pressure measurement in computer animation.

Our work uses a novel front end: a foot pressure sensor pad. The advantage of the foot pressure sensor pad is that it is viewpoint free (i.e., always under your feet) and occlusion free. The determination of foot-ground contacts, which has been proved very important to full body animation in previous work, is fully automatic. The pressure sensor pad does have its own limitations, which we will elaborate on later.

3.1.1 System Overview

FootSee consists of an off-line data capture and database construction stage (Section 3.3), an online motion recognition stage (Section 3.4), and a motion editing stage (Section 3.5). Figure 3.3 illustrates the main idea.

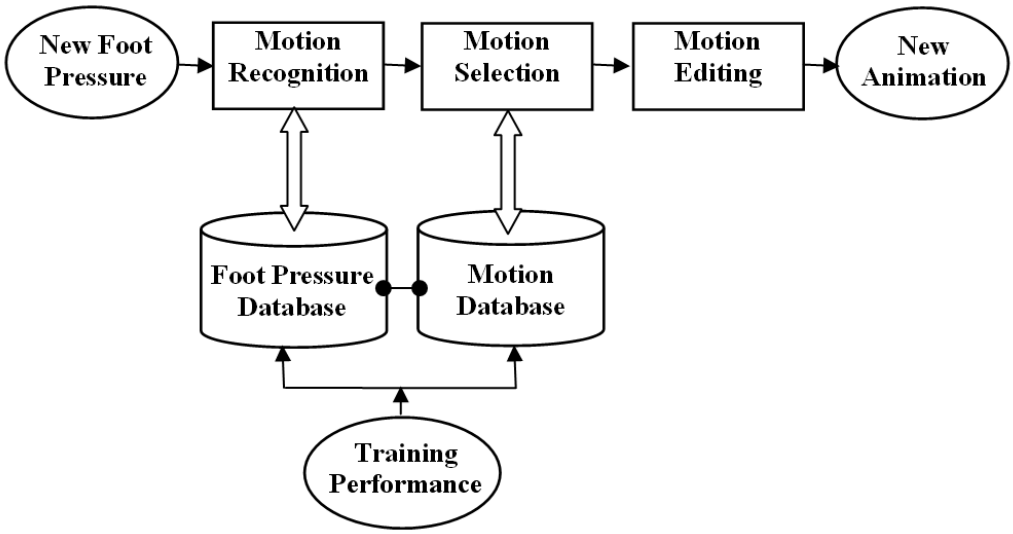


Figure 3.3: System illustration. The data in the foot pressure database and the motion database are properly synchronized during motion capture.

The motion database and the foot pressure database are properly synchronized during motion capture (see Appendix B.2), so that the motion selection module fetches

the motion corresponding to the matching footwork. However, we can also deliberately change the association rules and choose to map footwork to alternate motions. For example, we could map a marching-in-place motion to walking. This would allow the user to explore a virtual environment in a limited real environment.

3.2 Related Work

Work related to data-driven kinematic animation systems has been discussed in Section 2.1.1. Here we focus on intuitive animation interfaces.

A method of using footprints as a basis for generating animated locomotion was presented in [van de Panne 1997; Torkos and van de Panne 1998] . As the authors point out, appropriate timing and footprint positions are key to the success of their algorithms, but hard to obtain. A pressure sensor pad makes a good candidate as a front-end tool, replacing tedious and non-intuitive manual input. In the robotics community, however, there has been some work using bed pressure for monitoring patient activity, such as [Harada et al. 2001]. They target posture estimation while we aim at full body animation.

The idea of motion synthesis based on pre-captured motion databases has been extensively explored recently in [Kovar et al. 2002a; Arikan and Forsyth 2003; Li et al. 2002; Lee et al. 2002; Park et al. 2002a; Chai and Hodgins 2005]. State-of-the-art methods require a training or learning phase that typically lasts for several hours, after which, motions of a certain kind (locomotion, disco dance, or climbing) can be synthesized interactively, sometimes even in real time. In particular, [Lee et al. 2002] focused on interactive avatar control, and reported an intuitive vision-based interface that can control an avatar to step around a stool with a 3-second lag. Motion recognition was done by comparing visual features extracted from online images of the user and the pre-rendered images of the motions in the database. [Chai and Hodgins 2005] demonstrated an interesting performance driven system with two cameras and six markers. Vision-based interfaces have been studied in the computer vision community for quite some time for various tasks such as motion tracking and motion recognition [Moeslund and Granum 2001; Bottino and Laurentini 2001]. Our work is related to the overall ideas of [Lee et al. 2002] but with a novel front end: a foot pressure sensor pad.

3.3 Data Capture and Processing

3.3.1 Data Capture

The database consists of synchronized full body motion data and foot pressure data. Full body motions are captured by a Vicon 6 motion capture system at 60 Hz. Foot-

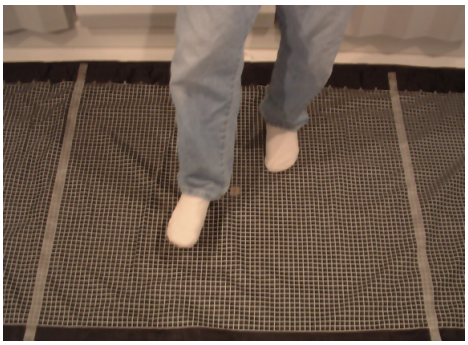


Figure 3.4: Custom XSensor pressure sensor pad.

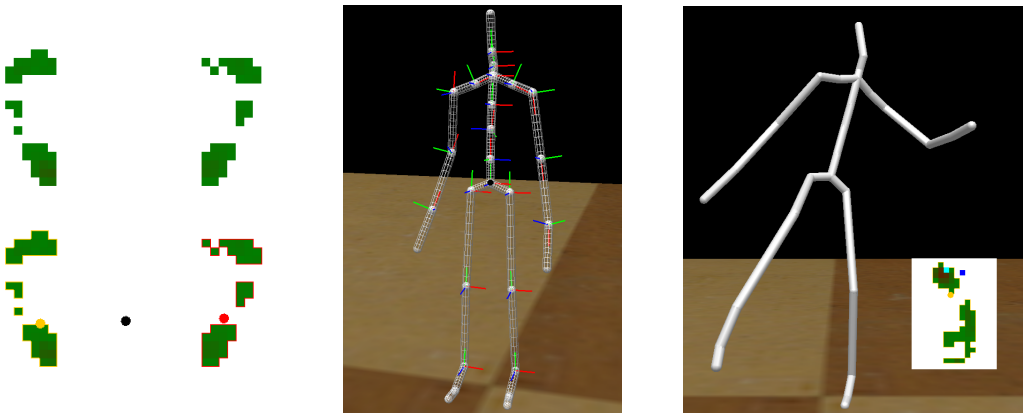
ground pressure data is captured at 12 Hz by an XSensor pressure pad, which is shown in Figure 3.4. This gives us five motion frames for every foot pressure frame. Our motion capture volume is approximately 1.5m long, 2m wide, and 2m tall. The pressure sensor pad is about 0.8m long and 2m wide. Given the constraints of the motion capture volume and the pressure pad dimension, we selected a number of behaviors to capture: quiescent stance with natural sway, and active motions including kicking, punching, stepping, weight transfer and balancing. The users were requested to return to quiescent stance between active motions. There was no specification for the duration of quiescent stance.

We use Filmbox[1] to map the Vicon marker position data onto the skeleton shown in Figure 3.5(b). This skeleton has 21 3-DoF (degrees of freedom) joints. The joint coordinates are represented by three axes (X, Y, Z), coloured red, green, and blue in the figure. The kinematic root shown as the black dot in Figure 3.5(b) is located at the intersection of the Lumbosacral angle of the spine, i.e., the base of the spine, and the pelvic girdle. Mapping the animation onto the skeleton converts the motion data from marker positions to root positions and joint angles. The root planar coordinates (the (x, z) coordinates) are further converted from absolute positions into relative positions (the difference between consecutive frames), so that transitions can easily be made between similar poses irrespective of planar location during later processing.

Figure 3.4 shows our XSensor pressure sensor pad. It is made of a 160 by 64 grid of pressure sensors. It was originally designed for measuring the pressures of a person lying on a bed, and was capable of sampling the whole pad at 6 Hz. Our pad is specially constructed to measure the larger pressures that arise from standing and running. Using custom software developed to our specifications, we can sample a smaller region of interest at a higher rate.

In our experiments we only sample the central 80 by 64 region to double the sampling rate to 12 Hz. The spatial separation of the pressure sensors is 0.5 inches. Each pressure pixel returns an 8-bit value representing 255 different pressure levels. The

[1]Kaydara Inc.: http://www.kaydara.com/



(a) (b) (c)

Figure 3.5: (a) A raw (top) and a labelled (bottom) foot pressure distribution image. Pressure level 1 is mapped to dark green and pressure level 255 is mapped to dark red. The black dot is the center of pressure. The coloured dots are the center of pressure of the same color bordered foot. The yellow foot is the left foot. The red foot is the right foot. (b) The skeleton model. The black dot is the kinematic root. The (X, Y, Z) axes of joint coordinates are represented by the red, green and blue axes. (c) A foot pressure image and its corresponding motion frame. The blue dot is the ankle position estimated from the avatar. The cyan dot is the ankle position estimated from the pressure data.

upper part of Figure 3.5(a) is a sample pressure image when the subject is standing still. For easy visualization, we discard 0 pressure values; we map minimum pressure 1 to dark green (R,G,B)=(0,127,0), and maximum pressure 255 to dark red (R,G,B)=(127,0,0). For quiescent stance, the pressure in all contact areas is generally low, so we only see green as shown in Figure 3.5(a). High pressure values usually show up in the heel area when there is only one foot in contact with the ground, as the red heel shown in Figure 3.5(c).

3.3.2 Feature Extraction

For motion recognition (Section 3.4), a 10-component feature vector $\mathbf{x} = (x_1, x_2, ...x_{10})$ is extracted from every pressure image. The feature vector consists of the velocity of the center of pressure (CoP) (2 components), contact area of both feet (2 components), and the first 6 Hu moments (6 components). The CoP velocity helps differentiate the motion directions (stepping direction, weight shifting direction etc.). The contact areas help differentiate motion types (one-foot-on-ground motion, e.g., kicking or stepping; or two-foot-on-ground motion) and bilateral motions (left-foot kicking or right-foot kicking). Hu moments are a set of algebraic invariants that com-

bine regular moments ([Belkasim et al. 1991]). They are invariant under change of size, translation, and rotation. Hu moments have been widely used in pattern recognition and have been proved successful in various applications. Other measures, such as separation of the feet, also come to mind as useful potential candidate feature. However, they are not used because they contain singularities. For example, when one foot is in the air, the foot separation is undefined from the pressure data alone.

We will later use the Mahalanobis distance ([Hastie et al. 2001]) of feature vectors as the distance metric for motion recognition. Assuming the component variables of the feature vector are uncorrelated, the Mahalanobis distance of $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$d_{i,j} = (\mathbf{x}_i - \mathbf{x}_j)^T C^{-1} (\mathbf{x}_i - \mathbf{x}_j) \tag{3.1}$$

where $C = diag\{\sigma_1^2, ..., \sigma_{10}^2\}$ is the diagonal covariance matrix computed from the database feature vectors.

To compute the contact areas of the left foot and the right foot, we first need to know the foot to which a pressure pixel belongs. In our setup, the Vicon cameras are all placed in front of the subject and all the motions we capture are facing in approximately the same direction. With the foot orientation known, foot tracking and recognition is very simple. The peak pressure point and all its neighborhood pixels within a bounding box (a box little bigger than the subject's foot) are classified as one foot. The peak pressure point of the remaining pixels and its neighborhood pixels are classified as the other foot. It is easy to determine which foot is left and which is right since the orientation of the subject is relatively fixed. In case there is only one foot in contact with the ground, as is the case for Figure 3.5(c), we determine left and right by comparing the current locations to the last known locations. For visualization, the left foot is outlined in yellow, and the right foot is outlined in red. The black dot is the center of pressure. The coloured dots are the center of pressure of the corresponding foot.

To perform inverse kinematics later (Section 3.5.1), we also need to estimate the ankle position of the user from the pressure images. When the foot is in full contact with the ground, i.e., when its bounding box is sufficiently large, the centroid of the last three rows of the foot is taken as the ankle position. The ankle position of the avatar is estimated by projecting the ankle joint position onto the horizontal plane, and then transforming it into the pressure pad coordinates. The transformation between the Vicon coordinate frame and the pressure pad coordinate frame is known at the capture and calibration stage. Usually we align the axes of the pad and Vicon calibration tools to reduce this transformation to a simple translation, determined by the placement of the Vicon calibration tool. In Figure 3.5(c), the cyan dot is the estimated user ankle position, and the blue dot is the estimated avatar ankle position.

3.4 Motion Recognition

The first step of the online motion synthesis is to recognize the user's motion by comparing the current foot-ground pressure image with the images contained in the foot pressure database, as illustrated in Figure 3.3. We note that there is no camera involved in the avatar control process once the database is constructed. The system "sees" the users through their feet. Our online motion recognition is based on activity detection and template matching. An off-line supervised learning process is carried out after the data capture. We describe it here rather than in the previous section because of its direct relevance to the motion recognition task.

Once the data is processed as described in the previous section, we manually segment the motion database into interleaved quiescent stance and active motion segments. This involves a human observing the motions in the database and marking where each active motion starts and ends. After the motion sequence is segmented, the proper segmentation of the pressure image sequence is also known because it is captured synchronously with the motion data. We then compute the mean μ and variance σ^2 of the features of the quiescent stance pressure images, i.e., we represent them as a random variable with a multidimensional Gaussian distribution $N(\mu, \sigma^2)$. For new input foot pressure data, we extract the features online as before (Section 3.3.2). We define an *activity score* s_i for frame i as the Mahalanobis distance from the feature vector $\mathbf{x}$ to the quiescent stance cluster center μ. When s_i is large enough (above a chosen threshold), we conclude there is an interesting motion possibly going on.

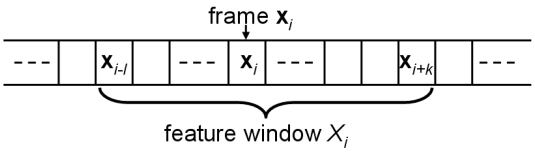


Figure 3.6: The composition of the feature window X_i for frame $\mathbf{x}_i$.

If a possible activity is detected, the feature vector of the current pressure frame i along with those of k look-ahead frames and l look-back frames are grouped together as a feature window X for the current motion: $X_i = (\mathbf{x}_{i-l}, ...\mathbf{x}_i, ..., \mathbf{x}_{i+k})$, as shown in Figure 3.6. The feature windows of the onsets of all the interesting motions in the database (denoted as $X_{i'}$) are candidates that we will compare with the current feature window. We then define the *motion distance* of two feature windows as

$$D_{i,i'} = \sum_{j=-l}^{k} w_j d_{i+j,i'+j} \tag{3.2}$$

where $d_{i,i'}$ is defined in Equation 3.1, and w_j is the weight computed from a simple linear hat function centered at $j = 0$, i.e., $w_j = 1 + j/(l+1)$ for $j <= 0$, $w_j =$

$1 - j/(k+1)$ for $j >= 0$, as shown in Figure 3.7. The motion $X_{i'}$ with the minimum motion distance to the current feature window X_i is recognized as the matching motion, and is selected for motion editing in the next step. To reduce false positives, i.e., a quiescent stance recognized as an active motion, a special quiescent stance motion is constructed by repeating μ for $(l + k + 1)$ frames, i.e., $X_o = (\mu, ...\mu, ..., \mu)$. If this quiescent stance motion is the best matching motion, the current frame is considered to be quiescent stance with a large sway, and no transition will be made.

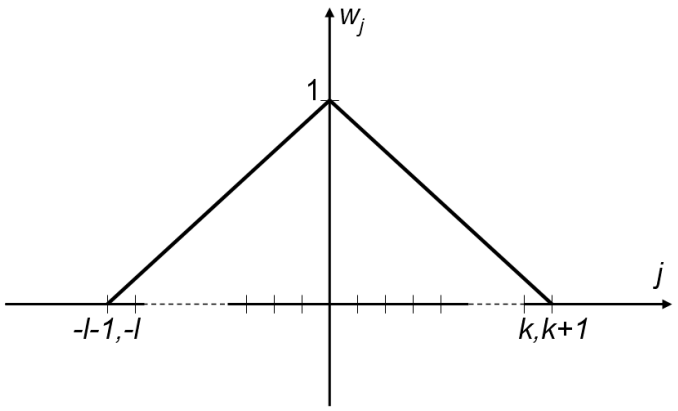


Figure 3.7: The linear hat function for weights used in computing motion distance.

During an active motion, we monitor the motion distance between the current input feature window and that of the recognized motion, rather than simply playing back the most similar motion to the end. The reason is twofold. First, similar behaviors may have different durations. This is especially true for the weight-transfer motions in our experiments. The user may shift their weight to a specific location (e.g., left leg), then remain in that pose for a while, and then shift the weight back, each time with different intermediate durations. If we find strong evidence that the selected motion is too slow or too fast compared to the user input, we perform timewarping (Section 3.5.3) to speed up or slow down the motion selected from the database, in order to be comparable to the currently observed motion. A second reason we monitor the motion distance is to belatedly identify mismatches. In such a case we need to perform a new search to identify a better motion. We always identify a good timewarp before giving up on the current motion and jumping to a new motion. An important point is that we cannot judge the quality of the current best match by $D_{i,i'}$, but rather we normalize it and use $D_{i,i'}/(s_i + s_{i'})$ as an indication of the quality of the best match. This is because for highly dynamic motions (motions with high activity scores) even a very good match may have large distance to the user motion, while a relatively subtle motion may have small distance to a bad match.

In our experiments, we use $k = 10$ look-ahead frames and $l = 6$ look-back frames. We initialize a motion buffer with 5 motion frames (the duration of 1 pressure frame). Subsequent synthesized motions are placed into the buffer at 60 Hz. All the transition

decisions plus the synthesis of the first motion frame after a transition can be made before the motion buffer underflows. The total latency of the system is 12 pressure frames: 1 frame of buffering, 10 look ahead frames, 1 current frame. This corresponds to one second at our pressure sampling rate of 12 Hz. We note that the recognition algorithm has a resolution of 12 Hz, the same frequency as the pressure data.

The current algorithm for organizing and searching data is sufficient for our test database. As the database grows larger, better algorithms may be needed. For instance, motion graphs techniques [Kovar et al. 2002a; Lee et al. 2002; Arikan and Forsyth 2003] may be used to construct transition locations. Hierarchical classification and organization of the motion database could also be exploited to help accelerate search speed [Lee et al. 2002].

3.5 Motion Editing

The motion recognition module identifies and outputs the motion in the database corresponding to the best-matching input pressure images. When there is a transition, either due to activity changes caused by user performance or by recognition mistakes, the current motion has to be smoothly blended into the new motion. Simple spherical linear interpolation of joint angle quaternions and displacement mapping [Bruderlin and Williams 1995] are currently used where applicable. Special manipulations, such as inverse kinematics and timewarping, are also used to maintain time and spatial constraints. One of the most important constraints to be satisfied is the foot-ground contact constraint. Because of the responsive requirements of performance animation, we choose algorithms that are as simple as possible.

3.5.1 Inverse Kinematics for Stepping

It is unlikely that the length and direction of a step taken at runtime will be exactly the same as those of the steps stored in the database. When the stepping error accumulates over time, the avatar could drift further and further away from the location of the real user. We developed an analytical inverse kinematics (IK) algorithm to modify the stepping motions selected from the database to exactly match the user's step length when a large position error is detected. The IK has the following steps, for which we will shortly give the rationale:

1. The root orientation and planar (x, z) position remain unchanged.

2. The hip joint angles are modified to meet stepping length changes.

3. After the hip joint angles are edited, some constraints will be violated, causing problems such as rotation of the foot during stance, and ground penetration

during swing. The ankle and knee joint angles are modified to keep these constraints satisfied.

The step length, step direction and step height are mainly a function of the orientations of the pelvic girdle, the left and the right hips, and the left and the right knees. In our experiments, the user is always facing the same direction, and so the orientation of the pelvic girdle (the root orientation) does not change much. We thus leave the pelvic orientation fixed in our IK. Steps 2 and 3 are justified by the fact that we consider only flat terrain in our experiments, and so the knee flexion-extension characteristics for different steps do not vary greatly. The hip angles are thus chosen to adapt the step length, and ankle and knee angles are used to maintain constraints.

Based on the above observations, we designed a fast IK algorithm detailed in Appendix C. When the step length change needed is small, it can be achieved in one stepping motion; when the change needed is large, we distribute the correction into several steps. Other IK algorithms could be used [Kovar et al. 2002b; Lee and Shin 1999]. Example-based IK may also be highly suitable [Rose et al. 2001].

3.5.2 Foot-ground Contact Satisfaction

The foot-ground contact is a hard constraint that needs to be satisfied at all times and treated explicitly whenever an edit may change the foot location and orientation. Simple blending between two stance poses that have the feet placed differently will result in foot sliding from one position to the other. Foot slippage and excessive rotation are visually very disturbing.[2] There are many techniques for dealing with this problem, which is also known as *footskate*. Some methods only allow transitions during changes of contact [Lee et al. 2002], while others have addressed this problem more generally [Kovar et al. 2002b].

We distinguish between two cases according to whether one or both feet remain in contact with the ground during a transition. Suppose we want to transition from motion frame i to frame j. If during the transition only one foot keeps in contact with the ground, the solution is simple. We treat the stance foot as the kinematic root. Only the stance ankle needs to be kept fixed and all the other joints are simply blended over time to the target configuration.

If both feet are in contact with the ground during the transition, we have two constraints that we have to maintain. The displacements of the lower-body configuration between frame i and frame j, including lower-body rotations and root translations, denoted Δ_{ij}, are computed. The configurations of frame j and its subsequent frames are then displaced by Δ_{ij}. The original constraints are maintained without visible degradation of the motion quality. Δ's of subsequent transitions are composed to-

[2]This supports our hypothesis that foot-ground interaction is constantly regulating our full body motions, and justifies our interface from another perspective.

gether with the current displacements should they occur. Over time, the lower body displacements Δ will accumulate and cause visible artifacts. We smoothly reset these displacements to zero, whenever one foot-ground contact breaks, such as in kicking or stepping motions. If there is no kicking or stepping in the database, an optimization routine such as [Lee and Shin 1999] could be invoked regularly to find a configuration as close as possible to the target motions while maintaining the foot contact constraints.

3.5.3 Timewarping

The duration and target frame for timewarping is decided at the motion recognition phase (Section 3.4). If we want to generate n motion frames for m frames fetched from the database, we need to slow down ($n > m$) or speed up ($n < m$) the ongoing motion. We need to resample the original joint angle trajectories. Define $j_i = \lfloor \frac{i}{n} m \rfloor$, and $\alpha_i = \frac{i}{n} m - j_i$. Then the ith synthesized frame should be the $(\frac{i}{n}m)$th frame in the original motion sequence, which we estimate by a linear interpolation of frame j_i and frame $j_i + 1$ with interpolation coefficient α_i.

Because the location of the root link is recorded in terms of displacement relative to the previous frame (Section 3.3.1), we need to accumulate them properly instead of simply resampling from surrounding frames. For this we use $\mathbf{p'}_i$ as the relative root (x, z) position for frame i:

$$\mathbf{p'}_i = \alpha_{i-1}\mathbf{p'}_{j_{i-1}+1} + \sum_{k=j_{i-1}+2}^{j_i-1} \mathbf{p'}_k + (1-\alpha_i)\mathbf{p'}_{j_i} \tag{3.3}$$

when $n < m$. The same idea is used to compute $\mathbf{p'}_i$ when $n > m$.

If the latency requirement can be lifted, for instance, in an off-line performance-driven animation system, we would have global knowledge of the motion, including knowledge of all upcoming motion, and thus perform a true dynamic timewarping algorithm, such as that in [Bruderlin and Williams 1995].

3.6 Results

Currently FootSee can control an avatar with a fixed latency of one second and render it at 60 Hz, on a dual-CPU (1.78 GHz Intel) machine. The CPU usage is under 10% during a typical session, leaving the CPU mostly free for other tasks such as dynamic simulation and game AI. We employ hardware rendering techniques to off-load the graphics rendering from CPU to GPU.

Figure 3.8 shows example frames from animations generated by FootSee. The lower left corner of each image shows the input foot pressure image. The lower right corner shows the corresponding pressure image of the best matching motion from the

database. The upper right corner is the controlled avatar. For comparison purposes, we also captured the motions of the real user. These can be seen in the upper left corner of each image. The motion of the user is rendered with a 1 second lag in order to be synchronized with the output of FootSee. More examples are available in the video http://www.cs.ubc.ca/∼kkyin/animation/Yin_SCA03.avi.

The database for this example consists of approximately 5.5 minutes of data. The motions are captured from a subject with no formal martial arts training. Each action typically lasts from 1 to 3 seconds. The active motions were performed randomly and repeatedly with 49 occurrences in total. We count a match as being correct if both the motion type and the motion extent of the user action and the avatar action match. Thus, a low punch matched to a high punch, or a right forward step matched to a right side step is counted as a mismatch. The recognition rate for four sessions with more than 10 minutes of motion is 80%. There are three typical types of errors. The first type is failure of recognizing upper body movement variations. For example, in Figure 3.8(l) a middle punch is matched to a high punch. The second type is confusion of low kicks and small steps. We found that the pressure images of the onsets of small kicks and steps are actually very similar. The third type is missing slow vertical weight transfers. The pressure images of slow downward weight transfer are sometimes very similar to those of quiescent stance. The same subject from whom we captured the database motions is used to test the system.

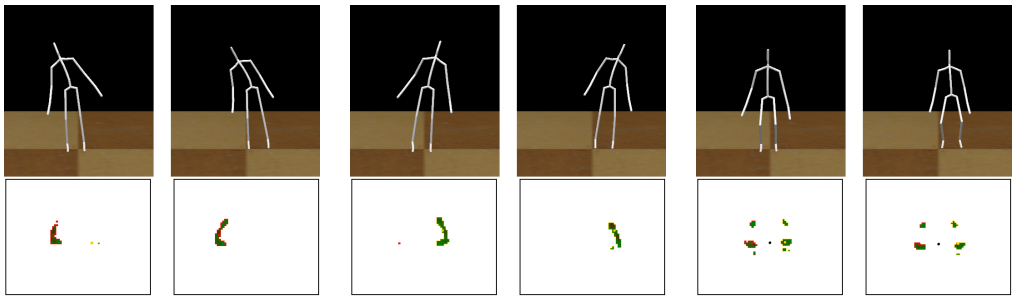
3.7 Discussion

In this chapter, we have developed a new interface for interactive avatar control and low-cost performance-driven animation, using a foot pressure sensor pad and pre-captured pressure and motion databases. It is intuitive, non-intrusive and reasonably robust. The novel use of foot pressure pads for performance animation points to possibilities of using other partial information for performance animation [Chai and Hodgins 2005].

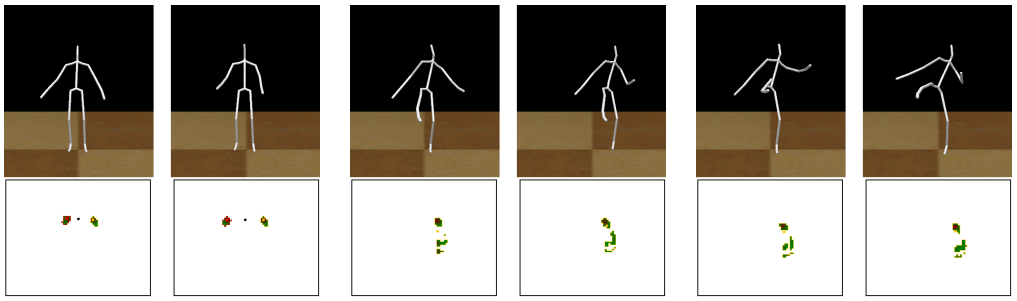
One cannot completely reconstruct all full body motions from foot pressure measurements alone, but we can get plausible motions using a well chosen database. The performance we have demonstrated may be sufficient for many applications such as virtual reality, video games, and performance-driven animation.

The system exhibits a trade-off between responsiveness and accuracy. Some recognition mistakes can be resolved if we allow for longer latency, and therefore make decisions using more information than onsets. Motion editing faces the same trade-off as well. If we know exactly where we are going before we step, the IK result will be better. In our situation, the leg may already be half way in the air before we see the stepping error.

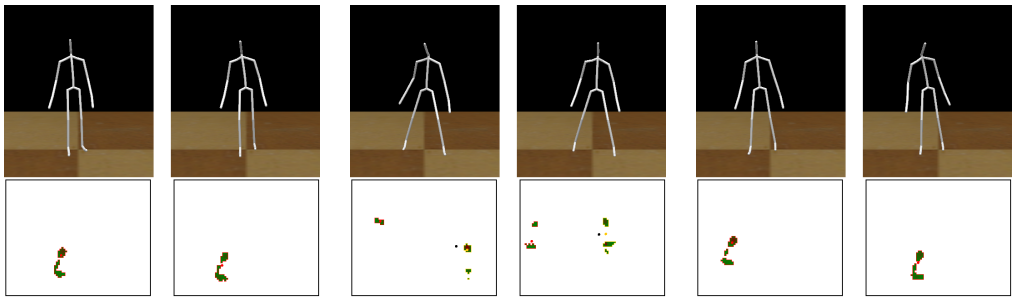
Where possible, we select simple, fast, and easy to implement algorithms, despite



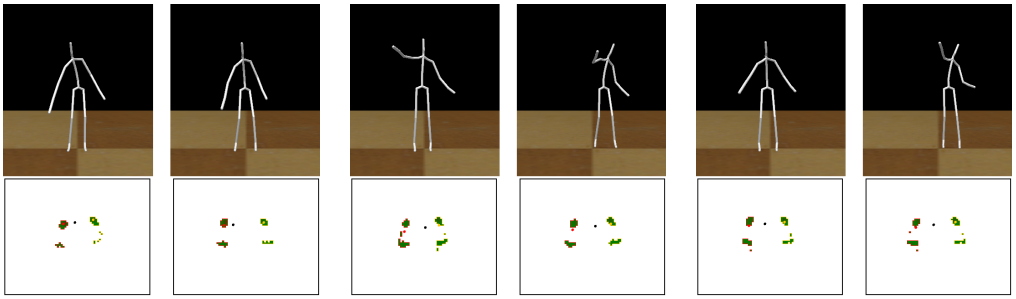
(a) rightward weight transfer (b) leftward weight transfer (c) downward weight transfer



(d) backward weight transfer (e) middle kick (f) high kick



(g) forward stepping (h) rightward stepping (i) leftward stepping



(j) low punch (k) high punch (l) middle punch

Figure 3.8: Sample frames generated by FootSee. Within each image, the lower left shows the input foot pressure image. The lower right shows the corresponding pressure image of the best matching motion. The upper right shows the controlled avatar. For comparison purpose, we render the motion of the user in the upper left, in sync with FootSee.

the availability of more sophisticated ones. For instance, our motion recognition algorithm needs negligible training time and a very small amount of training data. It is interesting that the meta-analysis of [Phillips and Newton 2002] suggests that more complex recognition algorithms do not necessarily work better. Our goal was to quickly develop a practical system to test this approach to performance animation, and to add complexity only when existing techniques fail.

3.8 Future Work

We would like to remove some of the current limitations in future work. The current foot tracking and recognition (Section 3.3.2) only works for a fixed facing direction. If the user changes his facing direction during database capture and online control, we may need to estimate the orientation of the feet from the pressure data as well. By carefully exploiting the foot spatial and temporal coherence, this may or may not be a hard problem.

Although currently FootSee is neither truly real-time nor highly accurate, it provides a solid point of departure. We would like to try higher pressure sensor density, higher pressure range and resolution, and higher capture rate, which should improve accuracy and reduce latency. We would also like to experiment with additional inference algorithms, and modelling of behavior dynamics to recognize motions and arbitrate ambiguities, for applications that can afford longer latency.

We would like to combine other non-intrusive sensing techniques with FootSee. For example, a camera can probably see the arm motions better than FootSee, while FootSee can resolve many ambiguities that a camera cannot.

Despite these current limitations, we believe FootSee is a promising technique that provides an intuitive and easy-to-use interactive interface for many types of applications, including interactive video games, avatar control, sports training, and performance-driven animation.

Chapter 4

Data-Driven Kinematic Animation of Interactive Balancing

A good animation interface equips users with the ability to control motion synthesis with ease. However, the ability to truly interact, such as being able to push a virtual avatar, is still lacking, as already stated in Chapter 2. Back-end algorithmic support is needed for the avatar to respond to users' input and recover from disturbances.

In this chapter, we describe a new data-driven approach for producing interactive dynamic balancing behaviors for an animated character. The result is a character that can interactively respond to single or multiple pushes in various directions and of varying magnitudes. Figure 4.1 provides an overview of the system. A database of captured responses to pushes is used to create a model that supports hip, arm, and stepping strategies for balance recovery. An interactive push is modelled as a force impulse, which is then used to compute a momentum-based motion index in order to select the most appropriate recovery motion from the database. The selected motion is then adapted in order to provide a response that is specifically tailored to the given force impulse while preserving the realism and style of the original motion. Based on a relatively small motion database, our system is effective in generating various interactive balancing behaviors, for single and multiple pushes. An example result is shown in Figure 4.2, where two forward steps are synthesized as a response to two pushes initiated by a user.

This chapter is structured as follows. Section 4.1 gives motivation and related work. Section 4.2 presents necessary background. For the actual system, we first construct a motion database (Section 4.3) which is populated with motion data from a series of experimental pushes applied to a human subject. In the online phase, a user can then interactively apply a push of a desired magnitude and direction. A motion selection algorithm (Section 4.4) then selects a matching motion from the database according to the momentum perturbation, as well as possible environmental or ZMP constraints. For example, a stepping strategy cannot be used when balancing on a beam. Once a motion is selected, it is adapted in order to more precisely match the

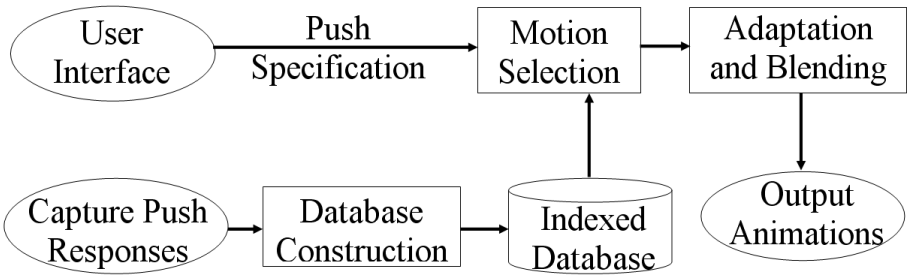


Figure 4.1: Kinematic balance system block diagram

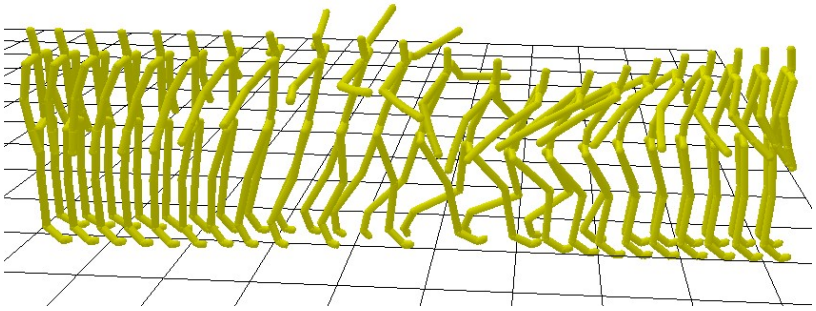


Figure 4.2: An example result. Two user-generated pushes (through a graphical user interface) result in two animated recovery steps.

current perturbation strength and direction. Finally, the result is blended with the ongoing motion in order to generate a plausible animation, while at the same time satisfying existing foot-ground contact constraints (Section 4.5). The algorithm is tested in Section 4.6.

4.1 Motivation and Related Work

Kinematic methods and dynamic methods each have their own merits as means for creating realistic interactive character animations. Kinematic methods driven by motion capture data offer realism while being limited in their generality. Methods that explicitly model dynamics have the promise of being more general, but require solving difficult control or optimization problems in order to deal with the active nature of human motion. In this chapter, we apply a data-driven approach to a restricted class of dynamic interactions with the environment, namely that of a character receiving unexpected pushes. We show that a data-driven approach to this problem yields a simple and effective solution for this case, and we expect that a similar methodology may be useful in constructing models of other types of dynamic interaction with the environment.

Human balance movements are classified as *semi-automatic* movements in the

movement sciences. These are non-trivial motor tasks that are learned in early childhood. Once acquired, humans can quickly select an appropriate motor plan from a repertoire of balancing motions in response to unexpected external perturbations. We develop an analogous data-driven approach for interactive balancing wherein a dynamic perturbation triggers the selection of a suitable balancing motion from a motion database, which is then further tailored for the particular perturbation.

Human balancing behaviors are of interest to a broad range of research areas, including biomechanics, robotics, and computer animation.

4.1.1 Biomechanics

The biomechanics and motor control community has been studying the human balance problem continually and extensively [Hoogvliet et al. 1997; Horak et al. 1997; Hsiao and Robinovitch 1998; Hsiao and Robinovitch 1999; Shumway-Cook and Woollacott 2001; Maki et al. 2003; Patla 2003]. However, a thorough understanding of the underlying neural control mechanism and musculoskeletal dynamic system has yet to be found. A common starting point has been the study of natural body sway and postural adjustments under small perturbations. For larger perturbations, various balancing strategies have been observed and analyzed, including the hip strategy, the arm rotation strategy, as well as change-of-support strategies such as stepping.

Two important concepts related to balance control are commonly used in the biomechanics literature. Humans regulate their body muscles to adjust the net Ground Reaction Force (GRF) between foot and ground to regain balance. The point of application of the GRF is constrained to be within the foot support polygon, and the GRF can only push but not pull the foot. The Center of Pressure (CoP) is defined as the point on the ground where the resultant normal GRF acts. It is mathematically equivalent to the Zero Moment Point (ZMP) (see Section 4.1.3) commonly used in the robotics literature [Goswami 1999].

4.1.2 Robotics

Biped balancing is a topic of considerable interest in robotics [Goswami 1999; Safonova et al. 2003; Goswami and Kallem 2004; Popovic et al. 2004; Abdallah and Goswami 2005; Komura et al. 2005b]. For humanoid robots, reference trajectory tracking based on regulation of various balance indices, such as momentum, ZMP, or CoM, forms the framework for the majority of the work [Kudoh et al. 2002; Lee et al. 2002; Ito et al. 2003; Kagami et al. 2000; Kajita et al. 2003a; Kajita et al. 2003b]. This framework is intended to deal with small perturbations caused by imperfections or by noise in the underlying dynamical system and the environment. Balance recovery from large perturbations remains a goal to be achieved. Our work in this chapter provides a solid starting point for an improved understanding of balance behaviors.

4.1.3 Computer animation

Balance is an essential feature that makes human character animations realistic. Spacetime optimization [Witkin and Kass 1988; Cohen 1992] and simulation methods [Hodgins et al. 1995; Pollard and Behmaram-Mosavat 2000; Faloutsos et al. 2001; Zordan and Hodgins 2002; Fang and Pollard 2003] emphasize the physical plausibility of motions, by incorporating dynamic constraints into the modelling and simulation procedure. These methods are usually computationally expensive and do not account for style. In contrast, motion editing techniques use measured motion data and emphasize rapid or interactive generation of new motions [Park et al. 2002b; Arikan and Forsyth 2003; Kovar et al. 2002a; Lee et al. 2002; Li et al. 2002; Arikan et al. 2003]. Recent developments are combining the advantages of both techniques, e.g., [Abe et al. 2004; Safonova et al. 2004], either to reduce the computation and increase the realism of synthesized motions, or to generalize motion capture data while preserving dynamic effects.

Recent results based on explicitly characterizing momentum patterns [Liu and Popovic 2002; Abe et al. 2004] are particularly relevant to our work in this chapter. To preserve the dynamic behavior of the input motion, [Liu and Popovic 2002] introduce a spline-based parameterization for the linear and angular momentum patterns of a captured motion. A family of similar motions are first optimized off-line, and then interpolated online to generate new motions. In this chapter, we also treat the momentum characteristics as a significant tool for modelling dynamic motions. We focus on applying fast and simple transformations that preserve the momentum characteristics as much as possible. Our approach works for momentum patterns such as the example shown in Figure 4.3, while the work of [Liu and Popovic 2002] uses a more constrained parameterization of momentum pattern tailored to a particular class of motion.

The Zero Moment Point (ZMP) is a balance index that is widely used in robotics, as discussed in Section 2.2.1. For physically plausible motions, the ZMP always remains inside, or on the boundary of, the support polygon, even when the character is falling. Key-framed animations and captured motions can violate the ZMP constraint due to errors and noise introduced during the modelling, capture and processing procedures. [Tak et al. 2000; Shin et al. 2003] each describe an interesting approach that modifies voluntary motions to make them satisfy the ZMP constraint exactly at all times. The reasoning behind using the ZMP in computer graphics is that while computer-generated motions may well be physically impossible, they look better when they are physically plausible.

Unlike previous work, we do not attempt to edit motions based on their ZMP constraints for several reasons. First, it is not obvious how to associate a computed ZMP pattern with the multimodal (hip, arm, stepping) balance strategies that we wish to support. Second, we do not know the perturbation forces for the balance-recovery

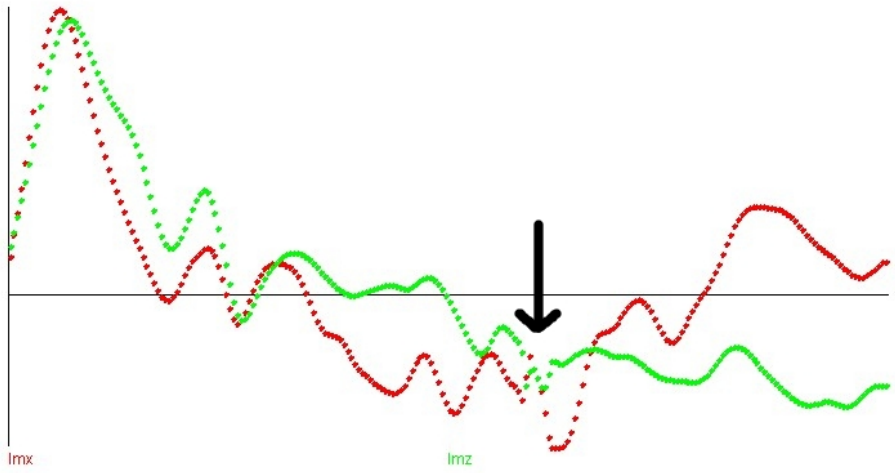


Figure 4.3: Total linear momentum of an arm-rotation balance strategy plotted as a function of time. The arms rotate three times before the subject recovers. The red and green curves are the X and Z components of the linear momentum. A Y-up right hand coordinate system is used. The broken segment labelled by the black arrow is due to motion capture noise.

motions that we capture, so we cannot precisely reconstruct the ZMP trajectories for these motions. Lastly, the computations for the ZMP are based on accelerations and are thus sensitive to noise in the captured data. In addition, it requires a model of the mass and inertial properties of the links, which may be difficult to estimate accurately. The issues in computing ZMP locations from motion data also arise in the work of [Tak et al. 2000; Shin et al. 2003], where terms are dropped from the ZMP equation in order to obtain a simpler and more robust estimate of the ZMP. In this chapter, we design transformations to respect ZMP constraints, rather than explicitly computing the ZMP.

Most recently, [Arikan et al. 2005] describe experiments with animation synthesis of pushing response. They parameterize push impulses as six dimensional vectors (three dimensions for the location on the body being pushed and three for the velocity of this location). Then a user-trained oracle selects a response motion from the database that will give the best visual quality when the necessary kinematic transitions and deformations are applied. They use a large database, captured from human push experiments consisting mainly of stepping balance responses, and can deal with unconstrained pushes on the upper body. Dynamic properties of the motions, such as the momentum characteristics and the ZMP constraint, are not directly taken into account.

4.2 Background

4.2.1 Balance Strategies

We capture hip, arm, and stepping balance strategies to populate our motion database. A *hip strategy* is characterized by body sway resembling a two link inverted pendulum by bending at the hip. This is typically elicited during perturbations that are large, on compliant support surfaces, or when the task requires a large or rapid shift in CoM. An *arm strategy* is characterized by rapid arm rotation. This is typically elicited during perturbations too large to recover by just lower limb strategies, and with environment constraints (i.e., standing on a ledge) or instructions that prevent stepping. A *stepping strategy* is characterized by asymmetrical loading and unloading of the legs to move the base of support under the falling CoM. This is typically elicited when there are no surface or instructional constraints, or when the perturbations are extremely large and in-place balance is not possible.

Although every strategy has its own kinematic and dynamic characteristics, they form a continuum of mixed strategies in the postural and balance response space, instead of being discrete strategies. In classifying the type of strategy that is observed in a given motion, we order the balance strategies as follows, from weak to strong: (1) hip strategy; (2) arm strategy; and (3) stepping strategy. We classify a motion according to the strongest balance strategy that is displayed. For example, if the subject rotated her arms as well as stepping, we label the motion as being a stepping strategy.

4.2.2 Momentum

For the captured motions that recover from pushes, we need to estimate the momentum perturbation injected by the external force. Linear and angular momentum (about the CoM), denoted as P and H_c, can be calculated as follows:

$$\begin{aligned} P &= \sum P_i = \sum m_i \mathbf{v}_i = m\mathbf{v} \\ H_c &= \sum H_i + \sum \mathbf{cc}_i \times P_i \\ H_i &= \mathsf{I}_i \omega_i \end{aligned} \tag{4.1}$$

where m is the total mass, $\mathbf{c}$ is the location of the CoM, $\mathbf{v}$ is the velocity of the CoM. Quantities with subscript i represent the quantity for just the i^{th} link of the rigid body system, e.g., ω_i is the angular velocity of the i^{th} segment, I_i is the moment of inertia of the i^{th} segment. The vector pointing from $\mathbf{c}$ to $\mathbf{c}_i$ is denoted $\mathbf{cc}_i$. We estimate mass-inertia parameters by approximating each limb of the motion capture subject with cylinders of matching sizes. We then estimate a uniform density to match the total mass of the virtual character with that of the motion capture subject.

The rate of change of linear and angular momentum can be calculated as follows:

$$\begin{aligned} \mathbf{f} + \mathbf{f}_{perturb} + m\mathbf{g} &= \dot{P} \\ \mathbf{cs} \times \mathbf{f}_{perturb} + \mathbf{cp} \times \mathbf{f} + (0, t_y, 0)^T &= \dot{H}_c \end{aligned} \tag{4.2}$$

where the external perturbation force $\mathbf{f}_{perturb}$ applies at point $\mathbf{s}$. $\mathbf{cs}$ is the vector pointing from the CoM $\mathbf{c}$ to $\mathbf{s}$. The resultant GRF $\mathbf{f}$ applies at the CoP $\mathbf{p}$, and t_y is the resultant moment around the vertical axis exerted by GRF. We only consider the instability about horizontal axes. That is, we do not model the rotation about the vertical axis that a strong off-axis push might introduce, such as a push to the right shoulder.

For all our captured balancing motions, we apply perturbations to a neutral stance pose. The pushes are applied at a consistent height and towards the central vertical axis of the character. We model the perturbations as impacts, that is, impulses of short duration that therefore do not allow the subject to have actively modulated the GRF during the course of the perturbation. During a perturbation, we thus assume that the GRF continues to go through the CoM and counteracting gravity, and therefore has no effect about horizontal axes for both $\dot{P}$ and $\dot{H}_c$. This allows for the following approximation:

$$\begin{aligned} \mathbf{f}_{perturb} &\cong \dot{P} \\ \mathbf{cs} \times \mathbf{f}_{perturb} &\cong \dot{H}_c. \end{aligned} \tag{4.3}$$

We can thus use Equation 4.1 to compute the maximum momentum of the system during a perturbation and its subsequent recovery. This provides an estimation of the momentum perturbation injected by the external force, with the assumptions that Equation 4.3 holds, and the balancer only removes momentum from the system.

4.2.3 Push Impulse Parametrization

We parameterize the push impulse and the corresponding captured balancing motion using the polar coordinate plot shown in Figure 4.4. The polar angle θ represents the perturbation direction. The polar radius r represents the perturbation magnitude. We estimate the direction and magnitude of perturbation impulses according to Equation 4.1. The color represents the balance strategy: hip strategies are shown in yellow; arm strategies are shown in orange; stepping strategies are pink. At runtime, users specify pushes to the virtual character by clicking on this plot.

4.3 Motion Capture and Database Construction

We captured six sessions of balancing motions, each being one to two minutes in duration, at 60Hz, using a Vicon optical motion capture system. The balancing

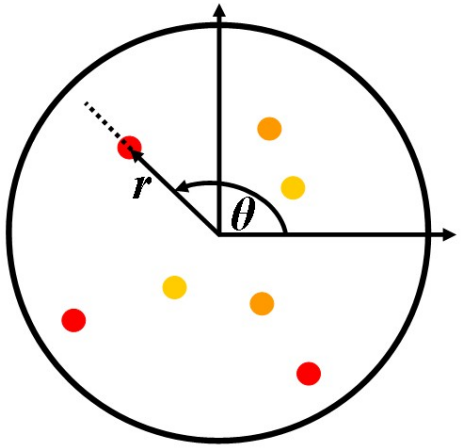


Figure 4.4: Parametrization of push impulses. This is representative of the GUI used to specify pushes at run time. A similar plot also serves as a visualization tool for the motion database, with the colour indicating the type of balance strategy that is invoked for a push of the given magnitude and direction.

subject was asked to balance naturally, and not to fake vigorous balancing motions in response to small perturbations. An assistant who served as a "pusher" was asked to push through the subject's central vertical axis, so as not to cause unnecessary momentum about the vertical axis. He was asked to deliver the pushes at a consistent height towards the middle of the torso, in order to make the relationship between the linear and angular momentum consistent across all perturbations. The pusher was requested to vary the magnitude and direction of the applied pushes. The pushes were to be with sufficient force to make the subject visibly move but sufficiently small so that the subject would never be close to falling.

We hand-segmented the resulting data into distinct balance-motion clips and hand-classified the balance strategy invoked by the subject. Our motion database consists of a total of 66 motion clips. The data is passed through a series of automatic post-processing stages. For stepping motions, foot-ground contacts and breaks are detected using a position/velocity threshold algorithm. The foot-ground contact information is used later by the motion selection and motion blending algorithm to favor plausible matches and smooth transitions.

We parameterize the push-response motions using the polar coordinates described earlier. For motions involving stepping, an interesting artifact is that a separate subject-controlled sideways shift of the momentum occurs in order to shift the center of mass (CoM) over the stance foot in preparation for stepping. Our momentum estimate produced by Equation 4.1 will therefore implicitly include this secondary effect, thereby potentially introducing an error in terms of estimating the magnitude and direction of the momentum perturbation applied by the push. Empirically and intuitively, the shift of CoM from the start to the end of the stepping conforms well

with the real perturbation direction. Hence, we choose the direction of the CoM shift as the polar angle for stepping strategies. The perturbation magnitude estimated from Eq. 4.1 is then projected onto this new direction. For consistency, we can apply the same direction and magnitude correction for in-place balance strategies as well (i.e., hip and arm strategies). The CoM shifts for in-place balance motions agree very well with the perturbation direction from Equation 4.1. Thus, there is no harm in applying this operation for in-place motions, while it improves the estimate for stepping motions.

Finally, we exploit symmetry to increase the number of motions in the database even further. A motion is mirrored with respect to the sagittal plane to produce its mirror motion. For all motions classified as being in-place balance strategies, we also interpolate a motion and its mirror motion in order to get a motion that is neutral with respect to the sagittal plane.

4.4 Motion Selection

Given a user-specified momentum perturbation and environmental constraints, we first employ a nearest neighbor (NN) algorithm to find the best matching balance motion for subsequent transformation and blending. We choose to use a NN algorithm over other potentially more sophisticated scattered data interpolation techniques because of two concerns. Firstly, interpolation does not work on push reactions using different balance strategies and inconsistent performances. Secondly, our motion database is sparse and sample motions are not evenly distributed.

Environmental constraints, such as that of a character standing on a balance beam can be used to further constrain the strategies that the character can adopt. We first enumerate the allowed strategies and then apply the NN algorithm among the motions that use the allowed strategies. We now describe in detail how the NN algorithm chooses a matching motion based on an input perturbation momentum.

For every pose Ω in the database, we define a maximum recoverable momentum $r(\Omega)$ that this pose can regain balance from, i.e., there is a trajectory in the database such that the character returned to a stable neutral pose successfully from this pose with momentum $r(\Omega)$. Usually we just define $r(\Omega)$ to be the actual momentum that this pose has in the database motion projected onto the perturbation direction as described previously. It is not necessarily the true upper bound for the recoverable momentum for this pose, but it is an estimate of what we can deal with given only the data found in the database. For every motion snippet, denote the initial pose as Ω_0, and the pose where the character reaches the maximum momentum as Ω_m, where 0 and m are the frame index for the first pose and the pose with maximum momentum. Then we set

$$r(\Omega_i) = r(\Omega_m) \qquad \text{for } i \in [0, m) \tag{4.4}$$

This modification says that all the frames before Ω_m are further away from the balance constraint boundary, so they should recover from at least momentum $r(\Omega_m)$. In case where we could deliver the perturbation instantly during our experiments, m would be zero and this operation would not be necessary.

In later motion adaptation process(Section 4.5), we only allow constraint-respecting motion transformations. For this purpose, we define a relationship operator $\succcurlyeq$ between two momenta such that $P_1 \succcurlyeq P_2$ denotes that the difference between their aligned polar radius is greater or equal to zero. Section 4.5.2 will describe in detail how we align momenta of different directions for different strategies.

We begin by defining the pose distance between two poses as $dp(\Omega, \Omega')$, which we take to be a weighted sum of the squared joint angle differences. Denote the distance between two momenta as $dm(P, P')$. We will take this to be a weighted sum of the squared differences of their polar coordinates, properly normalized and aligned.

$$dm(P, P') = (\frac{\Delta\theta}{\Delta\theta_{max}})^2 + k(\frac{\Delta r_a}{\Delta r_{max}})^2 \tag{4.5}$$

We take $\Delta\theta_{max} = \pi$, and Δr_{max} be the largest momentum in the motion database. Weight k is determined experimentally.

Given an input momentum P applied to pose Ω with momentum $m(\Omega)$, we wish to find a new pose Ω' by

$$\begin{aligned} \min_{\Omega'} \ & (w * dp(\Omega, \Omega') + dm(r(\Omega'), m(\Omega) + P)) \\ & s.t. \quad r(\Omega') \succcurlyeq m(\Omega) + P \end{aligned} \tag{4.6}$$

where weight w is determined experimentally. $m(\Omega)$ is defined as:

$$m(\Omega) = \begin{cases} 0 & \Omega = \Omega_0 \\ r(\Omega) & \text{otherwise} \end{cases} \tag{4.7}$$

Motion selection upon an initial push from a static neutral pose, and motion selection for a second response upon another push during the balancing of a first push are treated in almost the same way. The only difference is that in Equation 4.7 $m(\Omega) = 0$ when $\Omega = \Omega_0$. When the balancer is in the static neutral pose, the momentum should be zero because no push has been delivered yet.

If there are N motions in the database, and K frames for each motion, then a naive search algorithm for Ω' would be $O(NK)$. However, we can reduce the computation by pruning away entire motions. First, only a handful of motions in the database are close to the direction of $m(\Omega) + P$, and other motions need not to be considered at all. Second, if for a motion J of K_J frames, $r(\Omega_{J0}) \not\succcurlyeq m(\Omega) + P$, then motion J is immediately eliminated, since $r(\Omega_{J0}) \succcurlyeq r(\Omega_{Ji})$, for all $i \in [0, K_J]$. For a momentum perturbation applied to a neutral pose, the lookup is further simplified because the best matching pose defaults to Ω_{J0}, i.e., the 0th frame of motion J, due to the fact

that we captured all the motions from a common starting neutral pose. It is only for a second push occurring during the course of recovering from a first push that we need to search within a motion.

4.5 Motion Adaptation

After a best matching motion is selected from the motion database, we need to adapt this motion to make up the difference of momenta between the user input and the selected database perturbation. The basic principle is that we only transform the motion in the direction that poses less of a challenge to the underlying dynamic system, in terms of dynamic constraints (ZMP and torque limits) and kinematic constraints (joint angle limits). For example, if a motion is captured under a perturbation momentum ΔP, then an adaptation of this motion is very likely to handle a perturbation momentum $\Delta P/2$. In contrast, extrapolating this motion to recover from a perturbation momentum $2\Delta P$ may lead to implausible results, without formulating all the constraints into the algorithm explicitly. The $\succcurlyeq$ relationship operator defined in Section 4.4 allows us to check the validity of a transformation. A transformation is allowed when $P_{original} \succcurlyeq P_{new}$, which means the difference between the momenta's aligned polar radius, hereafter denoted as Δr_a, is greater or equal to zero.

We consider two types of motion transformations. The first is to adapt a motion to deal with a push from the same direction but with smaller magnitudes. We call this scaling adaptation. The second is to adapt a motion to deal with a push from a different direction. We refer to it as the rotation adaptation.

4.5.1 Scaling

If the perturbation magnitudes do not match, we apply a scaling transformation on the selected motion. We represent joint rotations using exponential maps $\mathbf{q}$, i.e., the rotation axis scaled by the rotation angle about the axis ([Grassia 1998]). Scaling of $\mathbf{q}$ by a scalar α amounts to scaling of the rotation magnitude.

$$\mathbf{q}' = \alpha\mathbf{q} \tag{4.8}$$

where $0 < \alpha \leq 1$, with corresponding scaling of the relative root joint displacements as well. We further apply an empirically-derived linear warp of the motion in time, observing that smaller perturbations allow a faster recovery. We can speed up the scaled down motions, ideally according to some space-time relationships for various balance strategies. We use a simple relationship that warps the time by $(1 + \epsilon/2)$ while scaling in space by $(1 - \epsilon)$. To achieve the original momentum scale factor, we solve for ϵ according to:

$$(1 - \epsilon) * (1 + \frac{\epsilon}{2}) = \alpha \tag{4.9}$$

Figure 4.5 shows the effects of the scaling operation on linear momentum.

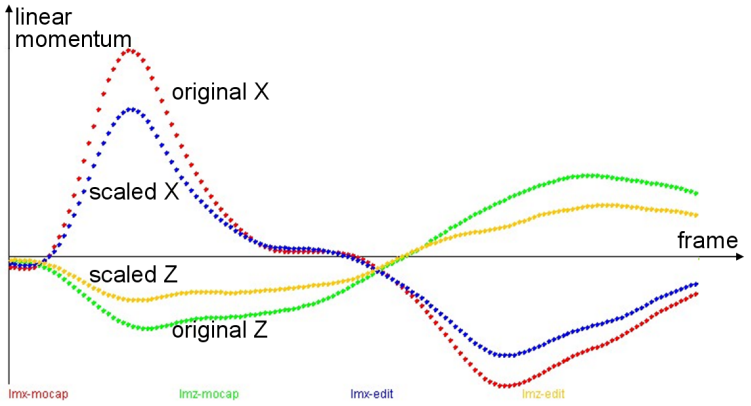


Figure 4.5: The X and Z components of the linear momentum with respect to time of the original (red and green) motion and the scaled motion (blue and yellow). Each dot represents one frame of the motion.

4.5.2 Rotation

If the direction of perturbations do not match, we perform a rotation transformation on the selected motion. If the perturbation is rotated about the vertical axis by an angle, the output motion and its momentum should also rotate. For example, if we push someone forward, she steps forward. If we push her to the left, then we expect her to step to the left. Of course we can push her in such a way that makes her to rotate to the left and then step forward. However as described in Section 4.3, we do not consider perturbation momentum around the vertical axis. Thus by rotation we do not refer to the character rotating her facing direction, but rather we refer to rotating the perturbation momentum and its corresponding balance recovery motion.

We apply separate algorithms for rotation of in-place balance recovery motions and stepping motions, as will be described shortly. Hence, the calculation for the aligned magnitude difference between two momenta Δr_a will also be different. Figure 4.6 shows an example of the effects of rotation operation on angular momentum. Linear momentum transforms similarly.

In-place rotation

For in-place balance motions, the foot support polygon is fixed and not radially equidistant. In Figure 4.7, if we want to rotate motion 1 towards the sagittal plane X, we interpolate motion 1 and its sagittal-plane-neutral motion 3. Given a new user input, the green triangle, its magnitude difference Δr_a from the database sample 1 is the distance marked by the bracket in the illustration: $\Delta r_a = \frac{r_1 \sin\theta_1}{\sin\theta_2} - r_2$, where r_1 is the momentum indicated by the orange dot 1, and r_2 represents the momentum of the green triangle.

The motivation for the projection of the original momentum onto the new direction

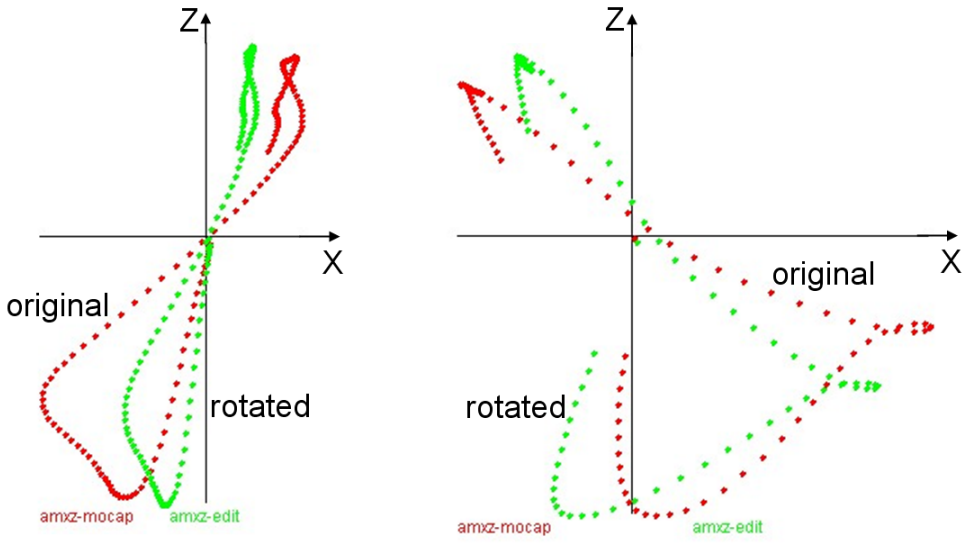


Figure 4.6: Angular momentum about CoM of the original (red) and rotated (green) motions. Horizontal axis is the X component. Vertical axis is the Z component. Each dot represents one frame of the motion. Left: in-place rotation. Right: stepping rotation.

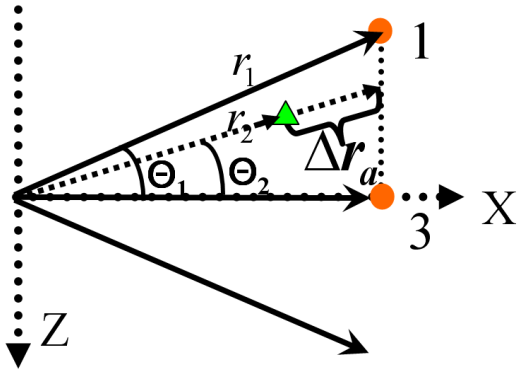


Figure 4.7: For in-place strategies, the momenta difference between the orange dot 1 and the triangle is Δr_a.

is as follows. Because we wish to respect constraints during motion adaptations, only rotations to less constrained directions are allowed. In Figure 4.8, direction 1 is the direction of the original motion with a shorter radius, and suppose now we wish to rotate it to direction 2. Since the support polygon has a longer extent in direction 2, the dynamical system has a larger stability margin and hence a longer time to remove perturbation momentum before it hits the ZMP boundary (i.e., foot support polygon boundary). Thus it is reasonable to say that the rotated new motion is physically valid given the same amount of perturbation momentum. However, we cannot justify the validity of a motion rotated from direction 2 to 1 because we may lose the ability

to recover balance due to the more constrained nature of the support polygon in this direction. If motion 2 already hits the boundary of various constraints, such as the ZMP constraint, then the rotated motion now in direction 1 will likely be implausible physically because the ZMP will shoot outside of the foot support polygon. That being said, we can still rotate a motion from direction 2 to 1, if we take into account the loss of recoverability when calculating Δr_a during our motion selection phase (Section 4.4), by properly projecting the database momentum onto direction 1.

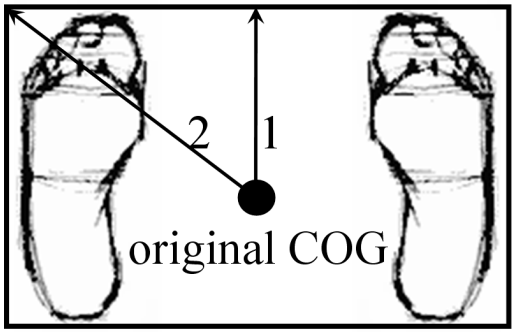


Figure 4.8: Foot support polygon.

Stepping rotation

For stepping motions, the goal is to change the stepping direction. The foot support polygon changes during the course of the motion. We thus do not need to project momentum for the rotation adaptation, and Δr_a would just be the difference between the polar radius of the two momenta: $\Delta r_a = r_1 - r_2$.

We denote the rotation about the vertical axis Y by matrix R, then

$$\mathbf{q} = R * \mathbf{q} \tag{4.10}$$

with corresponding rotation of the relative root joint displacements as well.

For single DoF joints such as the elbow, we keep the original joint rotations unchanged because the above operation will introduce additional degrees of freedom and render the transformed motion unrealistic when the rotation angle in R is large. For knee joints, such cancelling will move the foot position; we thus displace the hip joints accordingly to counteract such effects.

4.5.3 Blending

Selected and transformed motions from the database have to be blended with the current pose or motion in order to ensure a smooth transition. We use simple linear blending techniques together with simple root displacement techniques. Better

blending, foot skating elimination, and IK algorithms ([Rose et al. 1998; Kovar et al. 2002b]) could also be applied as necessary.

Certain constraints, such as foot-ground contact constraints, also have influence on the motion selection phase (Section 4.4). For example, a stepping pose Ω with the left foot in air cannot be easily blended to a stepping pose Ω' with the right foot in air. To disallow such blending, we set $dp(\Omega, \Omega') = \infty$ in Equation 4.6.

4.6 Results and Discussion

We tested our algorithm using a database consisting of 66 captured balance behaviors, shown in Figure 4.9. Users apply interactively perturbations by clicking in the circle of the GUI to specify the desired direction and magnitude, shown as the green arrow in Figure 4.9. The resulting selected and adapted balance motions along with the perturbations were then computed and animated in 3D in real-time.

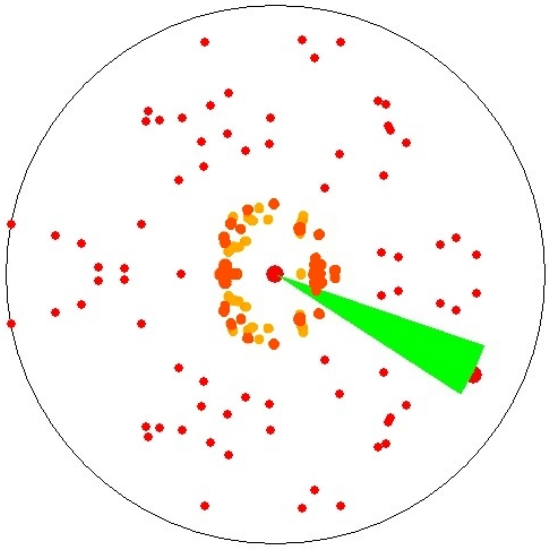
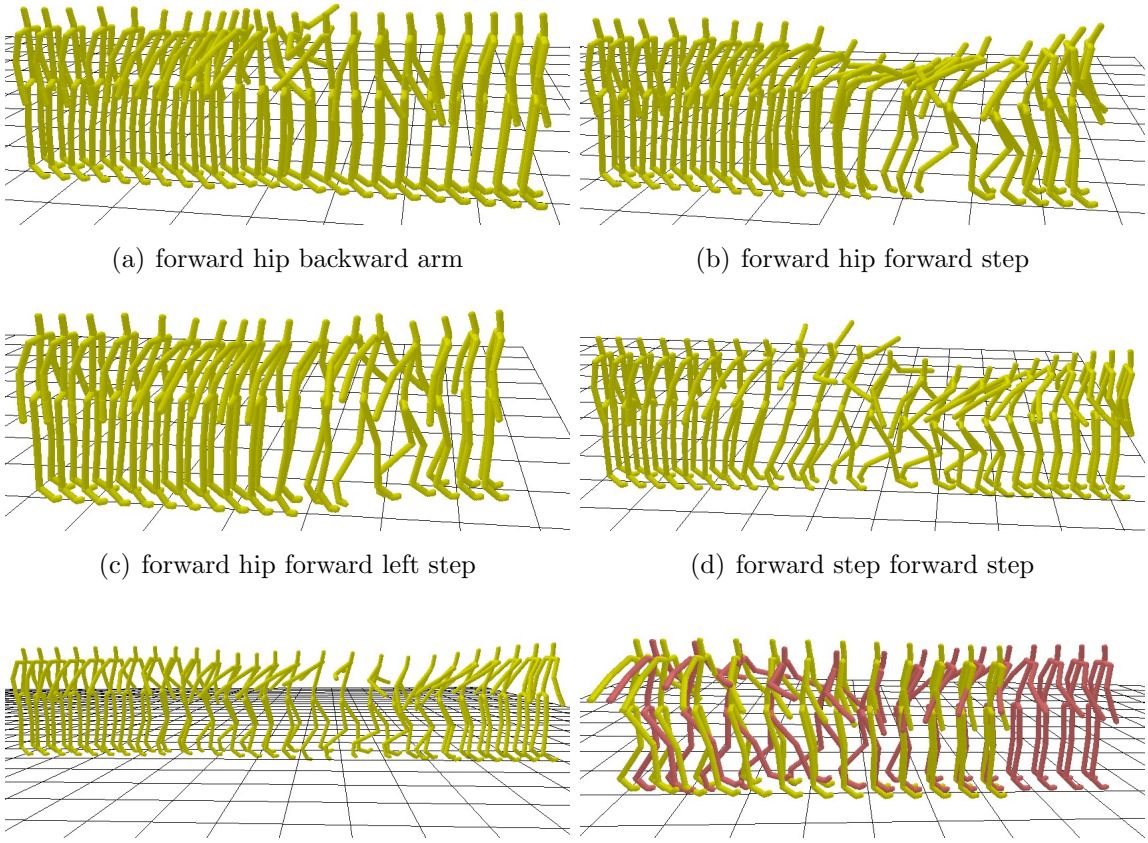


Figure 4.9: The motion database used for testing. The color scheme is the same as that of Figure 4.4. Balance recovery motions using hip strategies are shown in yellow; arm strategies are shown in orange; stepping strategies are pink.

Figure 4.10 shows a number of motions interactively generated from user inputs. We label each motion with the directions of the perturbations and the strategies of the matching database motions. For example, Figure 4.10(a) is a motion generated by a forward push then followed by a second backward push. The first matching motion is a balance response using hip strategy, the second matching motion is a balance response using arm strategy. Figure 4.10(f) is a backward stepping followed by a forward stepping. The forward stepping is shown in a different color and interlaced with the first backward stepping. More examples are available in the video



(a) forward hip backward arm

(b) forward hip forward step

(c) forward hip forward left step

(d) forward step forward step

(e) forward step forward step

(f) backward step forward step

Figure 4.10: Balance behaviors under perturbations of different directions and magnitudes. Each motion is labelled with the directions of the perturbations and the strategies of the matching database motions.

http://www.cs.ubc.ca/∼kkyin/animation/Yin_PG05.wmv.

Although our motion database only contains balance behaviors under single pushes, our system successfully generated motions that respond to multiple pushes, as well as to single pushes. This is important, because it is prohibitive to properly sample the space of all possible multiple-push scenarios.

Our system is fast and effective, even when using a relatively small motion database. It should be useful for interactive video games, fast balance behavior choreography, autonomous avatar control in virtual reality applications, and reference trajectory formation for humanoid robots.

In the future, we plan to investigate the other dimensions of the perturbation momentum we neglected in this work. For example, we wish to consider perturbations from different heights, and with an angular component around the vertical axis. We are also interested in studying responses to pushes during walking and running. The current blending of a response under a second push into the first response can

introduce physical implausibility, and this needs to be resolved for real robotics applications. More formal user evaluation of the results would be useful. We also plan to explore more interactive motor task synthesis using this kind of data-driven approach, with dynamic indices and constraint-respecting transformations.

Chapter 5

Dynamic Animation of Small Perturbations

Kinematic modelling is fundamentally limited by its kinematic nature. This limitation becomes increasingly apparent as we allow for richer user interactions, which will trigger various interesting dynamic behaviours. Beginning in this chapter, we focus on dynamic modelling of character animation. In this chapter, dynamic modelling refers to the full forward dynamics simulation of virtual characters. Our investigation will move from the simulation of small perturbations to the simulation of large perturbations (pushes). In this chapter, we will start by incorporating basic motor control principles to be able to simulate the effect of small external force perturbations applied to motion capture examples.

We propose a simple biologically inspired controller for motion perturbations. Motion capture is widely used for character animation. One of the major challenges of this technique is how to modify the captured motion in plausible ways. Previous work has focused on transformations based on kinematics and dynamics, but has not explicitly taken into account the emerging knowledge of how humans control their movement. We show how this can be done using a simple human neuromuscular control model.

Our model of muscle forces, as illustrated in Figure 5.1, includes a feedforward term, and a low-gain feedback term. The feedforward component is estimated from motion capture data using inverse dynamics. The feedback component generates reaction forces to unexpected external disturbances. The perturbed animation is then resynthesized using forward dynamics. This allows us to create animation where the character reacts to unexpected external forces in a natural way (e.g.,when the character is hit by a flying object) and still retain the quality of the captured motions. This technique is applicable to applications such as interactive sports video games. An example result is shown in Figure 5.2, where the captured motion of a football player responds to the impact of a ball on the arms.

We first outline the motivation for explicitly taking into account motor control in

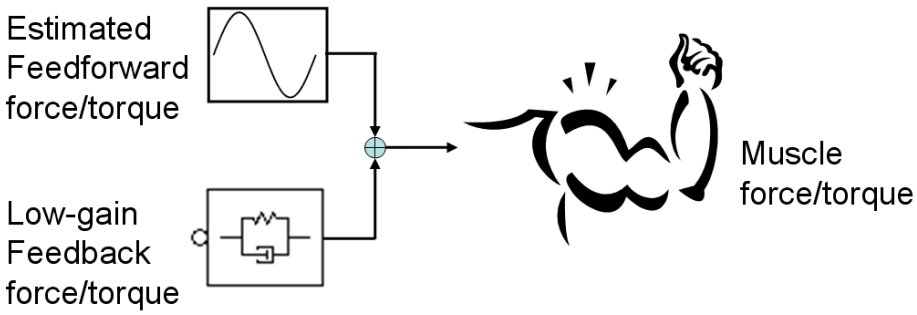


Figure 5.1: Our model of the muscle forces.

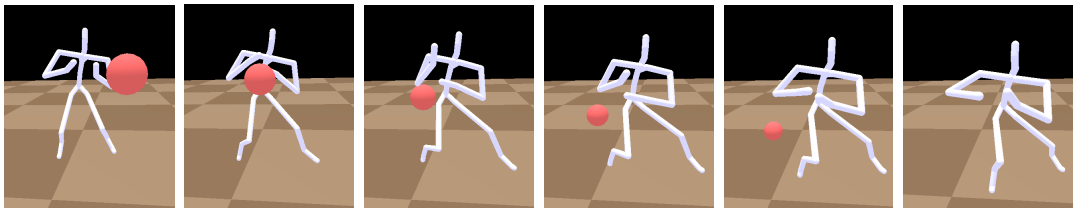


Figure 5.2: An example result. A captured motion responds to a ball hit.

a simulation framework in Section 5.1. Closely related work follows in Section 5.2. A simple human neuromuscular control model is then proposed in Section 5.3. We describe our dynamics simulation framework in Section 5.4. In Section 5.5 we describe how the dynamic system is coupled with the proposed motor control model, i.e., how the feedforward torques are estimated from motion capture data, and how the feedback torques are computed during the forward simulation. Section 5.6 demonstrates that captured motions can be made to respond to unexpected small perturbations in a natural way. Finally, a discussion of limitations and possible future work is given in Section 5.7.

5.1 Motivation for Motion Perturbation

Motion capture is increasing in popularity for realistic and stylistic human-like figure animation. However, the amount of motion that can affordably be captured is limited. In applications such as sports video games, the lack of variation between similar motions, or the lack of changes due to novel situations greatly reduces the sense of realism.

Generalizing motion capture examples using kinematic motion editing is relatively cheap, but ignores dynamic constraints. Dynamic motion transformations can guarantee physical plausibility, and interactions between characters and their environment can be handled naturally. However, dynamic human simulation still suffers from be-

ing unrealistic. The lack of realism may be attributed in large part to the lack of human motion control models.

The difficulty of the dynamic simulation approach is no longer the cost of the forward dynamics computations. Real time dynamic simulation of a humanoid is common [Kanehiro et al. 2004]. Rather the difficulty lies with creating realistic dynamic models and controllers. One approach to resolving the motor control problem is to manually design motor controllers for various motor skills. This is often tedious and difficult, and the resulting motion is often of low quality. Another approach is to use optimal control theory to solve for an optimal motion. This presents other problems. The goal of the optimization, as represented by an objective function, is often not clear. Empirical objectives are used, and there is no direct way to incorporate stylistic details. Optimization of a high dimensional nonlinear system is also computationally expensive.

It would be useful if at least a part of the control can be estimated from motion capture data. One way forward may be to explicitly take into account human motor control mechanisms for human character simulation. Ideally one would like to directly infer motor control mechanisms from motion capture data. Motion capture data encapsulates much knowledge of how humans control their movements, and also contains rich style information. We can also borrow from research in human motor control from neuroscience, biomechanics and other related movement sciences. Human motor control is still an active and contentious research topic. Nevertheless, even simplified models and general principles of human motor control can be useful in increasing the realism of computer animation.

We propose a way to incorporate a simple human neuromuscular control model into dynamic simulation systems. Original motion capture animations can be modified adaptively according to small unexpected disturbances rising from a dynamically changing environment. In this chapter, we focus on modifying upper body motions alone, and allow the legs to follow the original motion capture trajectories.

5.2 Related Work

5.2.1 Physics-based Animation

Skilled specialists have successfully designed motor controllers for dynamic human simulation by hand [Hodgins et al. 1995]. [Faloutsos et al. 2001] showed that such controllers are composable using machine learning techniques. Incorporating motion capture into dynamic simulations makes the control problem easier to solve, which in its simplest form is to directly use a tracking controller connected to a motion capture device as done by Zordan and Hodgins [1999]. Due to the similarity of this work to our own, we will differentiate them after discussing the related work from neuroscience

and movement science. Adjusting controller parameters results in force-based editing techniques [Pollard and Behmaram-Mosavat 2000].

Motor learning can be achieved through search. [van de Panne and Fiume 1993; Grzeszczuk and Terzopoulos 1995; Grzeszczuk et al. 1998] are able to synthesize motor control for basic motion tasks such as serpentine movement for fish, or locomotion for 2D characters for situations where balance is not a significant issue. The body configuration and optimization criteria are provided by the user. It is unclear how one could scale this technique to more complex systems such as humans. It is not straightforward to introduce explicit motion control knowledge into this framework, nor to model perturbation recovery from the environment.

5.2.2 Biologically Based Motor Control

A fact of biological motor systems is that neurons are slow when seen in the context of controlling movement. Sensory feedback through the periphery is delayed by a significant amount. For example, the delay in applying visual feedback to arm movements ranges from 150-250ms. Even a spinal reflex loop involving as few as three neurons can take on the order of 30-50ms. These are very large delays when compared with the total movement duration of very fast (150ms) to intermediate (500ms) movements [Kawato 1999]. Such delays can result in instability when trying to make rapid movements using high-gain feedback control. As a result, the high-gain feedback controllers which are widely used in robotics and control engineering are unrealistic for biological systems.

During the last decade, it has become increasingly accepted that the brain utilizes internal models of dynamics in planning and controlling motion (Section 2.3.2). The internal model theory proposes that the brain needs to acquire an inverse dynamics model of the object to be controlled through motor learning, after which motor control can be executed using feedforward muscle forces, in a largely open-loop manner [Kawato 1999; Mussa-Ivaldi 1999]. This explains why human movements show highly stereotyped and stylized patterns, although almost any task can in principle be achieved in infinitely many different ways. This also explains the observation that well-trained movements exhibit relatively low joint stiffness, i.e., changes in net torque at the joint due to displacements from the reference motion are small. During motor learning the stiffness is higher and the speed and accuracy are lower, due to the lack of good internal models [Gomi and Kawato 1997].

Internal dynamics models consist of *forward models* and *inverse models*. Forward models predict the behavior of the body and world with its knowledge about the body dynamics and environment. Inverse models invert the system by estimating the motor command which will cause a desired change in state. Forward models are important in motor planning in biological systems, since they can provide fast internal predictive feedback instead of relying only on the delayed feedback from the periphery. How

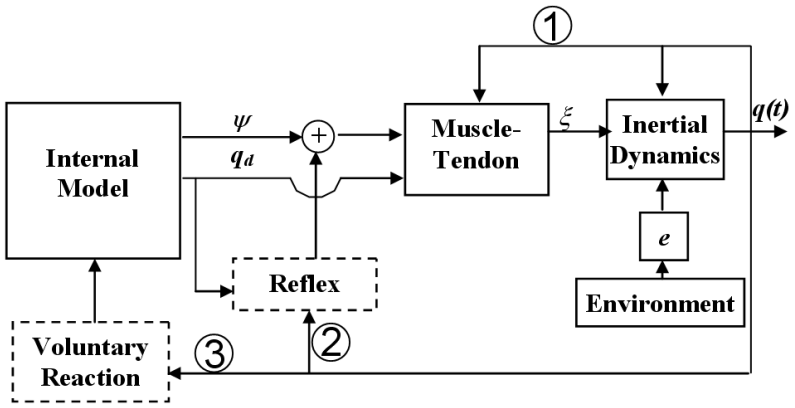


Figure 5.3: Our reference model of the motor system. Modules in dashed boxes are not implemented.

these models are used by the brain for planning motion is mostly unknown, and is currently the subject of active investigation [Harris and Wolpert 1998].

The feedforward neural commands from CNS internal models will be corrupted when descending to the muscles. The noise is signal-dependent and is usually assumed to increase with the mean level of the signal [Harris and Wolpert 1998]. Signal-dependent noise well explains many well-known facts and models in movement science literature, such as Fitt's law, minimum-jerk theory, and minimal-variance theory. There is also noise in the execution of a motion. The actual limb positions will drift from the desired trajectory planned by the CNS. Feedback control thus needs to be incorporated.

The intrinsic mechanical properties of muscles and tendons produce proportional (stiffness) and derivative (viscosity) feedback forces without delay [Hall 1998]. Our model uses this muscle property as a low-gain feedback controller to stabilize the limb along the desired trajectory. The muscle force-length relationships can be quite complex, but it is well known that muscle stiffness increases with generated force [Wise and Shadmehr 2002]. A simple model of this non-linear relationship is the bilinear model of muscle impedance [Hogan 1990; Winters and Crago 2000]. This model implies that the effective stiffness is proportional to the neural input, and hence the generated muscle forces. We assume that muscle viscoelasticities also increase in a similar way and are small for well-trained movements.

5.3 Our Motor Control Model

Figure 5.3 shows our reference motor control model. The internal model is treated as a black box whose output is the feedforward motor command ψ and desired trajectory q_d. The muscle-tendon system is driven by the motor command and generates the force ξ. Muscle force and external force act upon the human inertial dynamic system

and produce the actual trajectory. There are three feedback paths: ① muscle-tendon feedback; ② reflexes; and ③ voluntary reactions.

The precise details of the internal models are not necessary for our purposes. It is sufficient to know the role of feedforward torques in generating human motion, and the use of low-gain feedback. Instead of computing the feedforward torques from complex internal models, we estimate them directly for a given motion from motion capture data (Section 5.5) by inverse dynamics.

Among the three feedback loops, we only implement the muscle-tendon feedback, which has essentially zero delay. Reflexes have >30 ms delay, and voluntary reactions involves cortical replanning and have even longer delay. Our assumption is that for short duration unexpected disturbances such as being hit by a ball, the brain has no time to complete the long latency feedback loops and replan the motion. The trajectory is mainly restored by the low-gain muscle-tendon feedback forces. Thus we omit the long latency feedback modules and only consider the muscle feedback module. The muscle feedback controller we use is a hard-wired, low-gain and signal-dependent feedback controller (Equation 5.6). For well-trained unperturbed motions, muscle feedback has low gain, as discussed in Section 5.2.2. We can estimate the muscle force ξ by applying inverse dynamics to the motion capture data, and use ξ as an approximation of feedforward command ψ (Section 5.5.2).

We can contrast our approach to that of [Zordan and Hodgins 2002], which solves a similar problem. [Zordan and Hodgins 2002] simulate motion capture-driven motions that hit and react. They use a high-gain tracking controller when there is no external impact. The high stiffness parameters make the simulations appear overly strong and inflexible when contact is made. Therefore, the gains of the affected joints have to be reduced explicitly to allow the dynamics of the impact to influence the motion in a natural manner. Their high-gain tracking controller is similar to what is used in robotics, while based on all the previous discussions, we know that biological systems use a feedforward controller to do most of the work, and only use low-gain feedbacks to deal with neuromuscular noise. Upon perturbation, muscle stiffnesses actually increase due to the stretch reflex (the most important and most studied spinal reflex, see Section 2.3.2). We base our work on the fundamental characteristics of biological systems. We use feedforward forces to do most of the work. The low-gain feedback needs only to correct for simulation drift. Upon perturbation, the simple muscle stiffness model can model reaction and restoration, without a manually designed gain-scheduling controller as used in [Zordan and Hodgins 2002]. Extensions to more accurate, complex and biologically correct motor control models are also easier with our approach. For example, we can relatively easily incorporate a stretch reflex by changing the current muscle stiffness model to that of [Perreault et al. 2000]. Finally, our system simulates the perturbations in real-time, while [Zordan and Hodgins 2002] was an off-line system at the time of publication.

In general, impacts and other disturbances to the upper body may require the

lower body motion to change as well, to maintain balance and posture. [Zordan and Hodgins 2002] demonstrate how the lower body can be controlled separately using a balance controller that maintains the CoM within the support polygon. The control schemes only work when both feet are on the ground. Zordan's approach cannot be adopted directly into our system. The motions we use are highly dynamic full body motions involving stepping. For such motions, maintaining balance in a human-like way that is close to the quality of the original motion capture data is an unsolved problem. We will investigate this problem in following chapters.

5.4 Dynamics Framework

To incorporate the above proposed control model into dynamic simulations, we develop our own dynamic simulator. This in-house simulator provides us significant flexibility. Our dynamics simulation uses a velocity based Newton-Euler type formulation in maximal coordinates. Hereafter we only give a brief description of this framework, and refer the readers to Appendix D for more details. A Lagrange multiplier approach is used for computing constraint forces, inspired by the work of [Baraff 1996]. We extend this approach by allowing the simulator to solve both forward and inverse dynamics problems. Contact and collision are handled as linear complementarity problem (LCP).

5.4.1 Equations of Motion

We formulate the equations of motion in a full-coordinate constraint-based matrix form, instead of in reduced (generalized) coordinates. In the Lagrange multiplier approach, the velocity of each body is parameterized by a full six coordinate representation. Each joint in an articulated body is represented by a constraint equation, which is a linear equation on the velocities of the bodies. For a system with many constraints and many bodies we construct one large jacobian matrix J, containing all of the constraint equations, and concatenate the velocities of all of the bodies into a single vector $\mathbf{v}$.

The row space of J is the space of constraint forces. Similarly we introduce an analogous matrix H whose row space is the space of all possible "muscle forces", or more precisely "joint torques", which the muscles surrounding a joint can apply to their neighboring bodies. Combining these forces and torques with the Newton-Euler equations of motion gives us the following matrix equation:

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T & -\mathsf{H}^T \\ \mathsf{J} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \lambda \\ \tau \end{bmatrix} = \begin{bmatrix} \mathbf{f}_x \\ 0 \end{bmatrix} \tag{5.1}$$

Here M is the mass-inertia matrix of the bodies in the system (a block diagonal matrix with each block corresponding to one rigid body link), $\mathbf{a}$ is the acceleration vector of the bodies, and vector $\mathbf{f}_x$ contains external forces such as gravity and perturbation forces.

Discretize Equation 5.1, we obtain:

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T & -\mathsf{H}^T \\ \mathsf{J} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ h\lambda \\ h\tau \end{bmatrix} = \begin{bmatrix} \mathsf{M}\mathbf{v}_t + h\mathbf{f}_x \\ \mathbf{c}_e \end{bmatrix} \tag{5.2}$$

where h is the time step size, $\mathbf{v}_t$ and $\mathbf{v}_{t+h}$ are the velocity of the rigid bodies at time t and $t+h$. To counteract drift at the joints due to numerical error, we use a Baumgarte stabilization scheme [Baumgarte 1972]. $\mathbf{c}_e$ stands for the equality constraint stabilization quantity.

Equation 5.2 is a unified expression that is true for both forward and inverse dynamics. We now rearrange this equation to reflect the known and unknown variables in these two types of problems.

5.4.2 Forward Dynamics

In forward dynamics with known control, the muscle force multipliers τ are known quantities. Moving $\mathsf{H}^T\tau$ to the right hand side of Equation 5.2 gives

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T \\ \mathsf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ h\lambda \end{bmatrix} = \begin{bmatrix} \mathsf{M}\mathbf{v}_t + h\mathbf{k} \\ \mathbf{c}_e \end{bmatrix} \tag{5.3}$$

where $\mathbf{k} = \mathbf{f}_x + \mathsf{H}^T\tau$. Solving this equation at each time step gives us the updated velocity of the system.

5.4.3 Inverse Dynamics

Inverse dynamics is the process of finding a set forces that explain a given motion. The inverse dynamics equations we solve are another form of Equation 5.1. We move $\mathsf{M}\mathbf{a}$ to the right hand side of the equation (because the acceleration is a known quantity). Assuming the constraint equations $\mathsf{J}\mathbf{a} = \mathbf{0}$ are satisfied by the given motion, we no longer need the second row of Equation 5.1. We are left with:

$$\begin{bmatrix} \mathsf{J}^T & \mathsf{H}^T \end{bmatrix} \begin{bmatrix} \lambda \\ \tau \end{bmatrix} = \mathsf{M}\mathbf{a} - \mathbf{f}_x \tag{5.4}$$

If we can estimate the mass properties of the bodies of our articulated figure, along with the accelerations of its component bodies, then Equation 5.4 can be solved to determine the muscle forces multipliers τ.

5.5 Integrating Motor Control with Dynamic Simulation

We have developed a general-purpose rigid body simulation system as described in the previous section. The simulator is fast enough to simulate the dynamics of our 54-DoF character in real time. We now show how to extend it to meet the requirements of motor control. The integration of our motor control method and dynamic simulation consists of two main components. The first is a preprocessing stage, where we use inverse dynamics to estimate the feedforward torques from the motion capture data. The second component, which happens during the dynamics simulation, is the calculation of the actual muscle torques, which are a combination of the precomputed feedforward torques and feedback torques. The feedback torques depend on the difference between the motion capture trajectories and the current state in the dynamic simulation. This difference can be either caused by simulation inaccuracies, or by dynamic perturbations from the environment.

5.5.1 Algorithm Summary

Our approach can be summarized by the following steps, the details of which then follow.

- Preprocessing: inverse dynamics.
  - Estimate the mass matrix M for rigid bodies which approximate the shape of the character.
  - Fit a smooth curve to the position data.
  - Sample the accelerations of the rigid bodies at the rate at which we wish to run the dynamic simulation.
  - For each sampled time step:
    - Compute J and H given the positions of the rigid bodies.
    - Solve the inverse dynamics equation to determine the muscle torque multipliers τ, and store them.
    - Store the current joint angles θ_d and joint velocities $\dot{\theta}_d$.
- For each step during forward simulation:
  - Compute the current joint angles θ and joint velocities $\dot{\theta}$.
  - Compute the feedback torques $\gamma_1, ..., \gamma_n$, using Equation 5.6. Compute the feedforward torques $\psi_1, ..., \psi_n$, using Equations 5.7.
  - Compute the total external force $\mathbf{e}$.

– Solve the dynamics Equation 5.3 to determine the state of the system at the next time step.

5.5.2 Inverse Dynamics Preprocessing

Before beginning our dynamic simulation, we estimate a set of feedforward muscle torque multipliers τ for each time step. Given the mass matrix M, the constraint jacobian J, the matrix H, the external force vector $\mathbf{e}$, and the acceleration $\mathbf{a}$, we can calculate τ using Equation 5.4.

We first estimate the accelerations of the rigid bodies in each frame by fitting a cubic spline to the position data, and then finding the second derivative of the curve.

One problem in evaluating Equation 5.4 is that the mass properties of the character's component rigid bodies are unknown. We deal with this by approximating the shape of the character with polyhedra and computing the mass matrix for these, assuming the density of water (the body's average density is reasonably close to that of water). Another option to consider for future work is to use the approach described by Shin et al. [2003].

Another problem is that we do not know the ground reaction forces for the full body motions that we have access to. For this reason, and because maintaining balance during highly athletic motions is an unsolved problem, we simply constrain the root joint to move along the motion capture path. We can thus omit the contact dynamics with the floor. For future work, we can consider capturing ZMP trajectories together with full body motions, using the pressure sensor pads described in Section B.1.

In order to compute J and H, we require a kinematic model of the character to tell us the location and type of each joint. In our experiments, we use the kinematic model shown in Figure 5.4(a).

Using Equation 5.4, we precompute feedforward torque multipliers for each of the frames in the animation before beginning the forward simulation. The total feedforward torque is given by $\mathsf{H}^T\tau$. The dimension of τ is the sum of the degrees of freedom, which we denote with n. For the next section, we need to break this down into smaller components $\tau_1, ..., \tau_n$, each of which corresponds to one degree of freedom of the joints.

$$\begin{aligned} \mathsf{H}^T\tau &= \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \dots & \mathbf{h}_n \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} \\ &= \mathbf{h_1}\tau_1 + \mathbf{h_2}\tau_2 + \ldots + \mathbf{h_n}\tau_n \end{aligned} \tag{5.5}$$

We store τ_i's instead of $\mathbf{h_i}\tau_i$'s, because the orientation of the joints with respect to the world, i.e., $\mathbf{h_i}$'s, may be different in forward simulation than in the original motion capture animation.

5.5.3 Combining Feedback and Feedforward Torques

The muscle torques applied during forward simulation are a combination of the feedforward torque (as described in the previous section) and the feedback torque, which models the muscle dynamics of our motor control model.

Individual feedback torques are calculated for each degree of freedom of all of the joints. Let $\theta_1, \theta_2, ...\theta_n$ be joint angles corresponding to each degree of freedom, and $\dot{\theta_1}, \dot{\theta_2}, ...\dot{\theta_n}$ be joint velocities. The joint angles $\theta_{d1}, \theta_{d2}, ...\theta_{dn}$ are the "desired" joint angles – the joint angles from the motion capture data.

The feedback torque compensates for small drift and disturbances during the simulation. It is given by

$$\gamma_i = \mathbf{h}_i|\tau_i|\Big(k_{s_i}(\theta_{di} - \theta_i) + k_{d_i}(\dot{\theta}_{di} - \dot{\theta}_i)\Big) \tag{5.6}$$

where k_{s_i} and k_{d_i} are the stiffness and damping constants for each DoF. Partial measurements of these parameters are available (e.g., [Latash and Zatsiorsky 1993]), but complete data for humans is still unavailable to our knowledge. However, we do not require high accuracy for animation applications. We experimentally determine only one pair of these constants, and scale other gains according to the moment of inertia of the chain of bodies affected by that joint [Zordan and Hodgins 2002]. Note that the stiffness and damping gains in Equation 5.6 are scaled by the magnitude of the muscle torque multiplier $|\tau_i|$, as observed empirically in muscle biomechanics and discussed in Section 5.2.2.

The feedforward torques are computed by

$$\psi_i = \mathbf{h_i}\tau_i \tag{5.7}$$

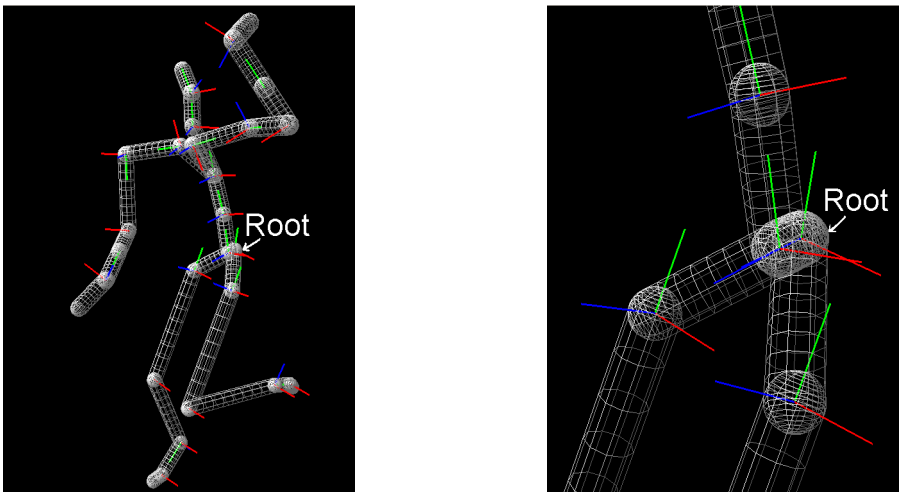
Our total muscle torques are given by the sum of the feedforward and feedback torques:

$$\xi = \sum_{i=1}^{n}(\gamma_i + \psi_i) \tag{5.8}$$

The muscle torques are added into the dynamics equation along with any other external forces, such as gravity and perturbations.

5.6 Results

Figure 5.4(a) shows the skeleton model we use in our simulation and rendering. It has 54 rotational DoFs in total. Figure 5.4(b) is a close-up of the kinematic root. It is a joint approximately located at the Lumbosacral angle of the spine. The dynamic simulation runs in real-time on a dual-CPU (1.78 GHz Intel) machine when there is no contact, or when there are simple contacts. The frame rate drops somewhat when complex contacts happen.



(a) (b)

Figure 5.4: (a) The skeleton model used in the character simulation. Each sphere represents a rotational joint. The total rotational degrees of freedom is 54. Red, green and blue axes represent the X, Y, Z joint rotation axes when the corresponding DoF is present. (b) A close-up of the kinematic root.

We perform perturbation experiments on captured arm motions as well as full body motions from football games. Figure 5.5 shows motion perturbation on an arm motion sequence. Figure 5.6 shows motion perturbations on two sequences of full body motion. In both cases, the simulated skeleton responds to external disturbances and returns to the original motion in a natural fashion. More examples are available in the video http://www.cs.ubc.ca/~kkyin/animation/Yin_PG03.avi.

For all the experiments, dynamic simulation is used even during times when there is no perturbation. In a real application where computation cycles should be saved whenever possible, a hybrid kinematic and dynamic system is preferable [Zordan et al. 2005]. That is, when there is no perturbation, motion examples can be played back kinematically. When a perturbation occurs, dynamic simulation is activated using initial conditions that are estimated from the kinematic trajectories.

5.7 Discussion and Future Work

Motor control is one of the major challenges in physically based human simulation. To our knowledge, our work is the first that tries to address this problem by explicitly incorporating human neuromotor control models into the human simulation system. We test our approach with motion perturbation tasks on motion capture data, and the results are promising.

The principal limitation of this work is that the implemented motor control model is very simple. Spinal and supraspinal feedback pathways are missing. This limits us to modelling short duration disturbances that do not endanger balance, and that do not involve motion replanning. We plan to address the longer latency (and also more complex and less well understood) feedback pathways in our future work.

Despite the simplicity of the implemented motor control model, animation with realistic responses to small perturbations can be generated, in real time. Applications such as interactive video games can thus benefit from an enriched motion repertoire. We believe that using a biologically based motor control model is the ultimate way to solve control problems in the dynamic modelling of character animation. In the future, it would be interesting to implement more modules of the proposed computational motor control model, and apply them to more challenging motion transformation tasks, such as generating realistic balance recoveries, dodges and falls in a real-life interactive sports video game.

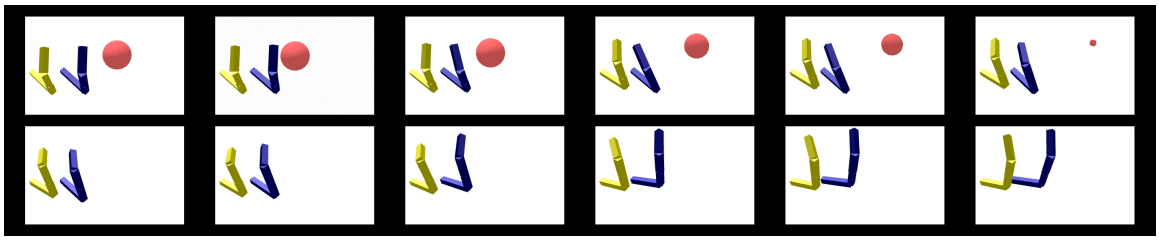


Figure 5.5: A simulated arm reacts to the impact of a ball hitting (between frames 2 and 3). The frames should be read from left to right and top to bottom. The arm, represented by 3 links, is fixed to the world at the shoulder (the lower end). The arm in the original motion is extending its elbow joint, and shown on the left in each frame for comparison.

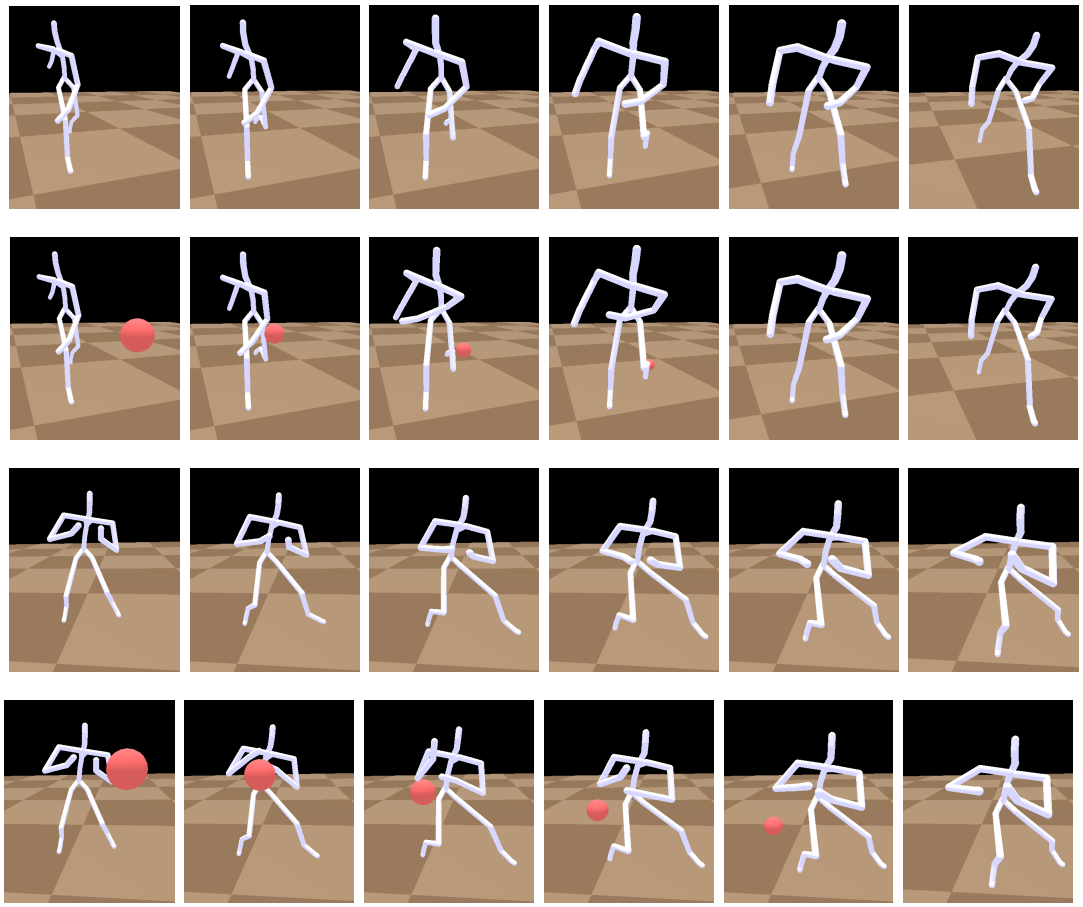


Figure 5.6: First and second row: an original motion and its perturbed motion, respectively. The impact occurs between frames 1 and 3. Third and fourth row: another captured motion and its perturbed motion, respectively. The impact occurs between frames 1 and 4. The ball has a radius of $10cm$ and a density of $1000kg/m^3$.

Chapter 6

Dynamic Animation of Interactive Balancing

We neglect the problem of balance in the previous chapter. In this chapter, we develop, integrate, and evaluate character balance controllers that, from an initial standing pose, can recover from unexpected external perturbations of varying magnitudes in arbitrary directions. This chapter can also be viewed as a dynamic parallel of Chapter 4. The supported balance strategies include ankle and hip strategies for in-place balance, single-step, double-step, and multi-step balance recovery. These strategies are further integrated with a walking controller. The limitations of each type of controller are mapped out in terms of the maximum push magnitudes and directions they can sustain. The controller construction can be informed using motion capture data if desired. Results are provided for a 30-DoF humanoid simulation that can be controlled at interactive rates. An example result is shown in Figure 6.1, where a double stepping controller constructed from a motion capture example recovers from a large push.

The control of balance for humanoids is an important and challenging problem. In this chapter, we focus specifically on the problem of balance recovery for small and large pushes during quiescent stance. The response to a push may come in many different forms, depending on the magnitude and direction of the push. Small pushes may require no stepping. Larger pushes may require a single step, double step, or possibly multiple steps before slowing to a stop. We present a system that integrates all of these strategies and document their performance limitations.

A particular contribution of this work is the omnidirectional nature of our controllers. Related work has most often focussed on modeling control in response to forward pushes in the sagittal plane. The control models we develop and demonstrate can respond to a wide range of push magnitudes and directions. We note that developing controllers for large lateral and diagonal pushes is a challenging task. The design of the stepping strategies can be tailored using motion capture data. Important parameters are precomputed in an offline optimization step in order to provide

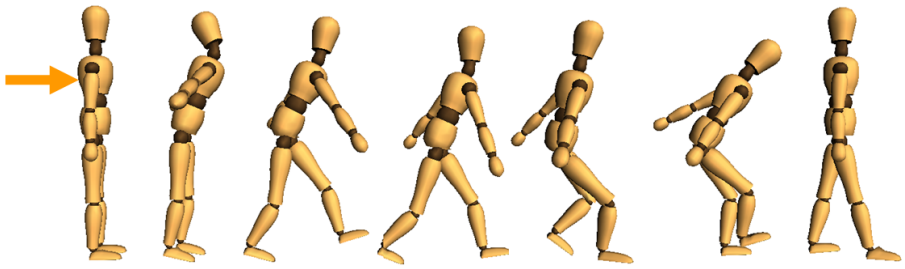


Figure 6.1: An example result. A double stepping controller reconstructed from motion capture data responds to a 600N, 0.2s forward push to the torso.

response in real-time.

The remainder of the chapter is structured as follows. Section 6.1 summarizes the closely related work. Section 6.2 describes the in-place balance controllers, including ankle strategy and hip strategy. Section 6.3 gives the details of the stepping controllers, including single-step, double-step, and multi-step strategies. Section 6.4 shows how to integrate various sub-controllers into a more powerful multi-strategy balance controller. We show the simulation results in Section 6.5. Lastly, Section 6.6 summarizes this chapter and discusses possible future directions.

6.1 Related Work

The literature in robotics, control, and biomechanics related to balance control for humans and humanoid robots is vast, and thus we necessarily restrict ourselves to a discussion of representative papers, as well as the specific work that is most related to ours.

6.1.1 Common Points of Reference

There are several balance measures and control mechanisms that are commonly used in much of the work in this area. In robotics, various balance measures have been developed, including: zero moment point (ZMP) [Vukobratovic and Juricic 1969], foot rotation indicator (FRI) [Goswami 1999], and zero rate of change of angular momentum (ZRAM) [Goswami and Kallem 2004]. With appropriate controller designs, these measures have been used to maintain balance for humanoid robots, most typically in the context of walking and relatively small disturbance [Kagami et al. 2000; Okumura et al. 2003]. Linear and angular momenta are commonly used quantities in motion and balance control. These measures are often used in combination with a reference trajectory that comes from motion capture data or an optimization procedure. Lastly, inverted pendulum models (IPMs) are often used as a tool to analytically

predict the motion of the center of mass during both quiescent stance and walking. For walking, IPMs can be used to help inform where the next foot-placement should be in order to retain balance during walking. More recently, the Capture Point and Capture Region have been proposed to predict when and where to take a step for a linear IPM augmented with a flywheel [Pratt et al. 2006].

6.1.2 Data-Driven Approaches to Balance Control

In recent years there has been considerable interest in using motion-captured reactions to pushes and generalizing these to new pushes that have not been previously observed. These techniques directly rely on a family of captured reactions and aim to interpolate and blend between them as necessary using momentum-based inverse kinematics and motion blending [Arikan et al. 2005; Yin et al. 2005; Komura et al. 2005a]. Although some try to take some dynamics into account, these techniques are kinematic. Thus while they may be adequate for purposes of computer animation, their use in interactively controlling the motion of fully dynamic humanoids remains unproven. See Chapter 4 for a detailed investigation.

6.1.3 Balance Without Stepping

The problem of in-place balance has seen a number of approaches, although they share many common elements. PD controllers based on CoM regulation using the ankles and hips have proven to be good models for sagittal plane balance control [Wooten 1998; Zordan and Hodgins 2002; Zhao and van de Panne 2005]. Recovery from large forward pushes during quiescent stance is an interesting case where a two-phase balance strategy becomes effective [Abdallah and Goswami 2005]. In the reflex phase, the body deliberately moves away from the ideal posture to absorb a disturbance force and maintain controllability. The recovery phase attempts to restore the body to its original posture as the disturbance force subsides. The motions do not involve any stepping. In [Kudoh et al. 2002], an optimization procedure based on quadratic programming is combined with PD control, which can boost the magnitude of pushes a humanoid robot can cope with. The focus is primarily on forward pushes in both [Abdallah and Goswami 2005] and [Kudoh et al. 2002].

6.1.4 Balance During Walking

The control of walking naturally involves issues of balance and has seen the use of many different approaches. Some representative examples with a particular focus on stepping control and balance are as follows. Seminal early work explores the use of a linear feedback strategy for swing-leg stride-length control to stabilize the dynamic walk of stilt-type bipeds [Miura and Shimoyama 1984]. In [Sugihara and Nakamura

2002], a balancing method with modification of pre-designed motion trajectories is presented. They use an Inverted Pendulum Model (IPM) to calculate desired CoG displacements upon disturbances, and then use the CoG Jacobian to coordinate the full body to recover the reference trajectory. Sensory reflexes, including a ZMP reflex, a landing-phase reflex, and a body-posture reflex, are used to stabilize a walk pattern generated off-line in [Huang and Nakamura 2005]. Feedback methods using the AMPM model (Angular Momentum inducing inverted Pendulum Model) have been proposed for bipeds to counteract external sagittal plane perturbations during walking [Komura et al. 2005b]. With this model, the walking steps are unchanged from the original feedforward motion. Limit cycle control, as described in [Laszlo et al. 1996], uses local linear return-map models to stabilize walking simulations onto a limit cycle. Our work relies on the limit cycle control strategy of Laszlo et al. [1996] for our multiple-step balance recovery controllers and our walking controller. We treat multiple-step recovery as a type of walking wherein the target speed is successively decreased over a sequence of steps until the character's translational momentum falls within the domain of our in-place controller.

6.1.5 Stepping Response Models

The stepping controller in [Kudoh et al. 2006] is the closest related work to our own. Essential parameters are first extracted from a sequence of motion captured human stepping examples to construct an appropriate inverted pendulum model (IPM). At the start of a stepping action, the trajectory of the CoM is determined by the IPM. Lower body motion is then generated by using inverse kinematic solutions for the support chain (stance ankle to CoM) as well as using IK to place the swing foot at a desired point along a target trajectory. A final optimization procedure addresses the dynamic consistency of the balance motion. Results are presented for significant forward disturbances in the sagittal plane during both quiescent stance and walking.

Like the work in [Kudoh et al. 2006], we employ a stepping strategy as necessary to deal with large perturbations. As in their work, we also integrate this with in-place balance strategies. However, our work employs specific response models for single-step, double-step, and multi-stepping strategies. We model the domains over which these various strategies can be enacted. Importantly, we test and document the performance of our controllers for perturbations from all possible directions. We thus address the challenge of designing controllers for large-magnitude diagonal pushes. For lateral pushes, our control model mimics the human strategy of stepping with the right leg for a push to the left. Although this at first seems counterintuitive, it reflects the fact that a push to the left will naturally unload the right leg, thus leaving it free to perform a cross-step to the left in order to retain balance. Our control strategies can be informed by motion capture data. In Section 6.5 we make specific performance comparisons of our controller with some of the controllers reported in the literature,

to the extent that this is possible.

6.1.6 Overview of Human Balance Strategies

The biomechanics and motor control community has studied human balance in considerable depth. Studies of how we control the equilibrium of the body in the face of gravity and environmental disturbances have led to the concept of *movement strategies* [Winter 1995; Horak et al. 1997; Shumway-Cook and Woollacott 2001; Maki et al. 2003; Patla 2003]. Postural strategies describe general sensorimotor solutions to the control of posture, including not only muscle synergies but also movement patterns, joint torques, and contact forces. From human subject experiments, various balancing strategies have been observed in response to external perturbations, including an ankle strategy, a hip strategy, as well as change-of-support strategies such as stepping.

The *ankle strategy* uses distal to proximal muscle activation, primarily at the ankle and the knee. It is characterized by body sway resembling a single-segment-inverted pendulum and is typically elicited during small shifts on flat support surfaces or perturbations of CoM when the task requires maintenance of upright posture.

The *hip strategy* uses early proximal hip and trunk muscle activation. It is characterized by body sway resembling a double-segment inverted pendulum divided at the hip. It is typically elicited during perturbations that are large combined with a lack of a surface to support a step, on compliant support surfaces, or when the task requires a large or rapid shift in CoM.

The *stepping strategy* uses early activation of hip abductors and ankle co-contraction. It is characterized by asymmetrical loading and unloading of the legs to move the base of support under the falling CoM. This is typically elicited when there are no surface or instructional constraints, or when the perturbations are extremely large and in-place balance is not possible. Multiple steps may occur during balance recovery.

6.2 In-place Controllers

We now describe our controllers. We begin with balance control without stepping, which is the most limited in terms of its ability to react to a push.

6.2.1 Ankle Strategy

Biomechanically, the ankle strategy is the process of using ankle plantarflexors and dorsiflexors, and invertors and evertors to adjust the CoP to provide the needed moment to push the perturbed CoM to the desired location. The ankle strategy can be viewed as an equilibrium point tracking mechanism. We implement it as a proportional derivative (PD) controller that produces a virtual force $\mathbf{f}$ regulating CoM

position $\mathbf{p}$ and velocity $\dot{\mathbf{p}}$.

$$\mathbf{f} = k_p(\mathbf{p}_{desired} - \mathbf{p}) - k_d\dot{\mathbf{p}} \tag{6.1}$$

The virtual torque needed from a joint (ankles, knees and hips) is defined by:

$$\tau_v = \mathbf{f} \times \mathbf{r} \tag{6.2}$$

where $\mathbf{r}$ is the vector from the CoM to the individual joint center. The virtual torque is then transformed by R, the matrix that relates the global coordinate system to the joint's coordinate system, into actual joint torques:

$$\tau_j = R\tau_v$$

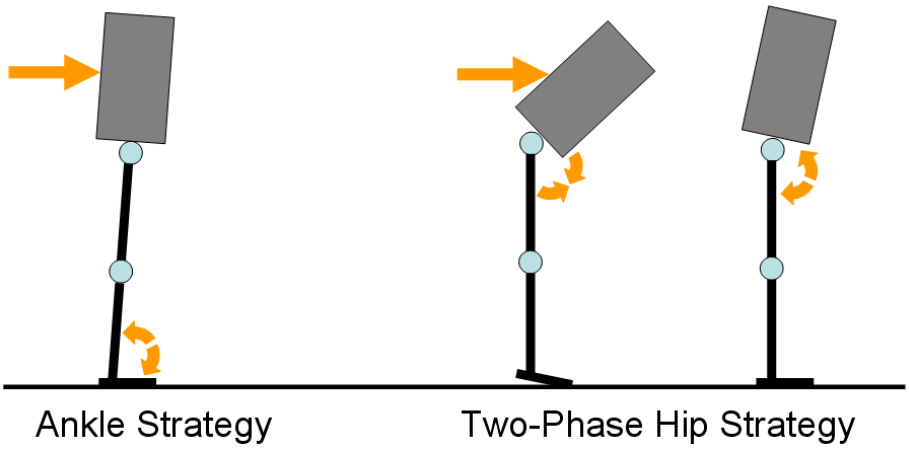


Figure 6.2: Left: Ankle strategy; Right: Hip strategy

6.2.2 Hip Strategy

Hip flexors/extensors and abductors/adductors are more effective in displacing the CoM. In addition, from Section 6.1 and [Abdallah and Goswami 2005], the rotation of the hip induces angular momentum which shifts the ZMP inward with respect to the foot support boundary. The hip strategy consists of a reflex phase and a recovery phase [Abdallah and Goswami 2005], as illustrated on the right of Figure 6.2.

We implement this mechanism as simple linear synergy that co-activates the hips based on ankle torques computed from the ankle strategy:

$$\tau_{vh} = s\tau_{va}$$

where τ_{vh} is the virtual hip torque, τ_{va} is the virtual ankle torque. In our experiments, we used $s = -3.0$. Due to kinematic constraints, the hip strategy is primarily effective

for forward pushes [Winter 1995], and thus we only apply the hip torque when the sagittal component of the joint torque is larger than $-50Nm$.

To facilitate the self-induced rotation, the hip position gain (i.e., k_p, see Equation 6.3) is lowered to 20% of its original value during the yielding phase, and regains its strength as a linear function of time over $\beta_1 = 2.0$ seconds during the recovery phase. The virtual character transitions to the recovery phase when the CoM velocity rotates by more than 90 degrees, i.e., $\dot{\mathbf{p}}_t \cdot \dot{\mathbf{p}}_0 <= 0$, where $\dot{\mathbf{p}}_0$ is the original CoM velocity, and $\dot{\mathbf{p}}_t$ is the current CoM velocity. This separates the hip strategy into two distinct phases.

6.3 Stepping Controllers

Pushes of large magnitude require a stepping strategy. We classify our stepping controllers into three types, based on how many steps the controller evokes before returning to a static upright posture.

- *Single step*: Only steps once, and recovers to an upright posture with a staggered foot stance.
- *double step*: Steps twice, and brings the trailing foot next to the stepping foot when returning to the upright posture.
- *multi-step*: Takes more than two steps to recover, with the number of steps being determined on the fly. The final quiescent stance pose is similar to the starting pose.

All of the stepping controllers are characterized by three phases:

- **Phase 1**: Change of support. In this phase, the controller decides where and how to lift-and-place the swing foot to change the foot support polygon. This is repeated multiple times for double-step and multi-step control.
- **Phase 2**: Reduction of momentum of the system. In this phase, the controller removes momentum either through stiffness and CoM velocity regulation in single and double stepping, or through up-vector regulation [Laszlo et al. 1996] in the case of multi stepping.
- **Phase 3**: Return to the upright posture. The controller steers the character back to an upright posture.

We now describe the various components of the stepping control in more detail.

6.3.1 Where and How to Step

The basic control representation used for stepping and walking is a pose control graph (PCG), which is a type of finite state machine as shown in Figure 6.3. Each state P_i specifies a set of desired joint angles for the character, and the joints are driven toward the desired angles through the use of PD controllers:

$$\tau = k_p(\theta_{desired} - \theta) - k_d\dot{\theta} \tag{6.3}$$

Figure 6.3 shows a typical PCG for stepping. It first lifts the left leg as shown on the left, and then steps down as shown on the right. State P_n can be the same as P_1 for a cyclic graph, usually for cyclic motions such as walking.

The state transition conditions in the PCG are time-based or sensor-based. In the PCG in Figure 6.3, a transition occurs from pose P_1 to pose P_2 after 0.3s. The sensor based transition conditions we use in this work include foot contact and CoM velocity. For example, in the PCG for multistepping and walking, $P2$ will transition to P_3, the symmetric counterpart of P_1, after the swing foot contacts the ground. CoM velocity is monitored to allow a transition to Phase 3 of the stepping controllers when this is deemed feasible based upon a velocity threshold. The PCG provides a base motion upon which control adjustments are then layered.

The PCG contains a small number of poses ($n = 4$) and can be designed either manually as in [Laszlo 1996], or they can be extracted from motion capture data. In order to extract key poses from motion capture data, we employ kinematic centroid segmentation [Jenkins and Mataric 2003] to extract the key 'leg lift' and 'foot placement' poses, which are used as poses P_1 and P_2 respectively. Additionally, we allow continuous tracking of the upper body motion data as a function of time in order to more precisely replicate a particular desired arm, torso, and head movements during a motion.

Given a base PCG, balance control is layered on top of it by modifying the control variables, i.e., target joint angles, of the swing hip, which thereby parameterizes the placement of the swing foot. This type of foot placement strategy is implicit in most inverted pendulum models used for human balance and walking [Winter 1995; Kajita et al. 2003a; Kudoh et al. 2006]. The two Euler angles of the swing hip are θ_s for the sagittal angle and θ_l for the lateral angle. The control variables for the stepping thus consist of $\mathbf{c} = (\theta_s, \theta_l)^T$. We compose the final desired hip angles by adding the control variable hip angles to the hip angles from the base PCG.

Given the above general control framework, a first key decision to be made is which leg should take the first step required to recover balance? Upon an unexpected push, it is most natural to step using the leg which is unloaded by the push [Maki et al. 1996]. More formally, we determine this using the sign of $\dot{\mathbf{p}} \cdot \mathbf{e}$, where $\dot{\mathbf{p}}$ is the CoM velocity at the beginning of a step, and $\mathbf{e}$ is the load line from the right foot to the left foot. A positive value implies a push with a component to the left, and thus will

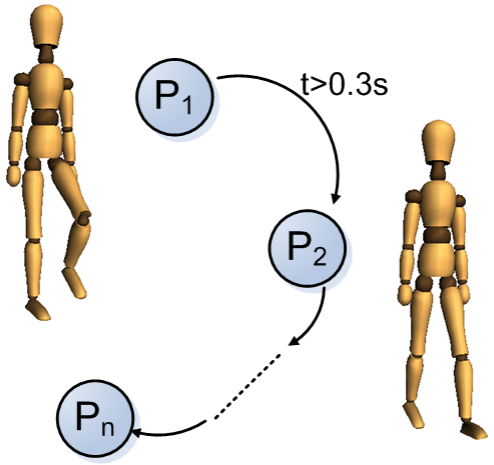


Figure 6.3: A pose control graph represents the basic control for stepping and walking, and is a finite state machine. The states P_i specify desired poses. State transitions are either time-based or sensor-based.

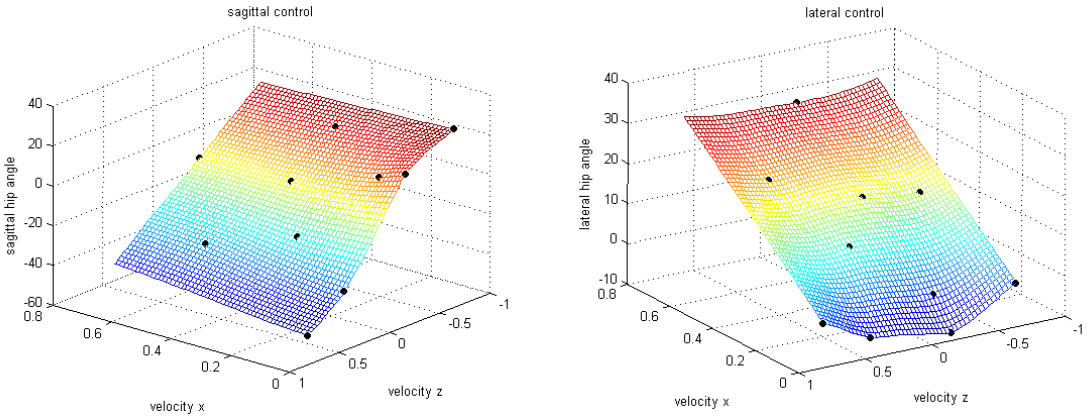
unload the right leg which should then be the leg to take a step.

A second problem to solve when pushed is that of determining *where* to step to help recover balance. This will typically be a function of the current motion of the CoM. As in [Laszlo et al. 1996] we apply an offline search procedure to compute solution families and then use a function approximator to replace the search procedure with an oracle that can then efficiently provide answers for online use.

The search procedure works in a simple and effective fashion as follows. A set of four simulations is carried out for each step, two of which sample the result of step sizes of $L + \Delta L$ and $L - \Delta L$, where L is a default step length in the sagittal plane. A further two simulations sample the effect of making lateral perturbations of $-\Delta W$ and ΔW to the default target step placement. For each simulation, a desired target quantity called the regulation variable (RV) is measured, thus allowing the construction of a simple finite-difference-based linear model of how variations in foot-placement affect the regulation variable, which is the quantity we wish to control. For single and double stepping controllers, the RV used is the final CoM velocity. For multi stepping and walking controllers, the RV consists of the trunk angles with respect to the vertical. The above description captures the spirit of the control strategy, and we refer the reader to [Laszlo et al. 1996] for additional details.

In order to make the stepping controllers operable in real time, we precompute solutions for a family of planar pushes that span a range of directions and magnitudes and then use a function approximator to build a predictive model. Given a set of planar pushes of different directions and magnitudes $\mathbf{f}_i = (f_{ix}, f_{iz})^T$, we record the planar CoM velocities $\mathbf{v}_i = (v_{ix}, v_{iz})^T$ as the state variable, right after the push ends and just before any balance controller starts to execute. We use a Y-up coordinate system, and the y component of the velocity is discarded. Using the search technique,

we record the controls $\mathbf{c}_i = (\theta_{is}, \theta_{il})^T$ required to minimize the final COM velocity for each example push. Using a small set of samples, we then construct a function $\mathbf{c}_i(\mathbf{v}_i)$ using a thin-spline function approximator. Figure 6.4 shows an example of the resulting control surface for a single-step controller. Given a new body state, the controls are readily predicted using the function approximator, which enables real-time online control.



(a) (b)

Figure 6.4: Thin plate interpolation of control points for single stepping. Left figure shows hip sagittal angle $\theta_s(\mathbf{v}_i)$; right figure shows hip lateral angle $\theta_l(\mathbf{v}_i)$.

6.3.2 Single and Double Stepping

Our single and double stepping controllers share the same mechanism for Phase2 and Phase3, after touchdown of the swing foot:

- Reduction of momentum of the system. This is achieved in two ways. First, the position gains for the swing foot and knee are dropped to zero upon ground contact in order to reduce impact, and regains its original values linearly over time in $\beta_2 = 0.5$ seconds. Second, active torques are added to regulate the CoM velocity, i.e., Equation 6.1 with only the damping term.

- Return to the upright posture. When the velocity of the CoM is below a threshold $\gamma = 0.15m/s$, we deem it is now safe to start to return to the desired upright posture. A kinematic planner calculates desired ankle and hip angles based on the current foot configuration and straightened knees. This new pose is added to the PCG to steer the character back to this posture over a duration of $\beta_3 = 0.5$ seconds.

These two phases follow the philosophy of Section 6.2.2 and [Abdallah and Goswami 2005]. Reduction of momentum and recover of the upright posture can be conflicting

goals in balance tasks upon large pushes. Separating them into two phases results in a more successful and robust balance controller.

6.3.3 Multi Stepping Controller

The single-step and double-step controllers both provide rapid stops, i.e., they bring the motion to rest as quickly as possible. Another strategy we investigate is a more gradual stop whereby the momentum caused by external forces is gradually dissipated. For this we employ the walking controller proposed in [Laszlo et al. 1996].

In [Laszlo et al. 1996], a PCG gives the basic open-loop controller for walking. Local linear models are then constructed through preview simulations to stabilize a set of regulation variables (RVs), and hence the walking itself, onto a limit cycle. Different open-loop controllers and different RVs can be used to achieve different styles as well as controlling the walking direction. For our purposes here, we only use a straight-line forward-walking controller, with the up-vector as the RVs. The up-vector is a vector aligned with the principal axis of the torso, and measures the torso lean in the sagittal and frontal planes. To begin a walk, the torso typically leans forwards. Upon being pushed forward, the torso will lean forward substantially. For each step, instead of commanding the feedback controller to regulate the up-vector about a fixed target value as in a cyclic walking controller, we decrease it over time with the goal of achieving a gradual stop:

$$RV_n = \alpha * RV_{n-1} \qquad\qquad 0 < \alpha < 1$$

where n is the counter for the number of steps. We set $\alpha = 0.9$. When the velocity of the CoM along the walking direction falls below a threshold $\gamma = 0.15m/s$, we execute Phase 3 as described in Section 6.3.2 for the single and double stepping in order to achieve a full stop and postural recovery. This approach can also cope well with multiple pushes sustained during the gait.

6.4 Multiple Strategy Integration

Given the in-place, single-step, double-step, and multi-step control strategies, we next develop a model that experimentally evaluates the capabilities of each of these strategies for pushes of arbitrary direction. These models provide a reference for comparing the performance of our implemented control strategies with other published performance data (where available). They also enable us to design a controller that integrates the various balance strategies in a robust way.

We record the maximum push $\mathbf{f}$ a controller can deal with in each of five chosen directions. We denote points for strategy S_i as $\mathbf{s}_{i_j}$, $j = 1, 2, 3, 4, 5$, and denote the convex hull of these points as $H(S_i)$. We only deal with half the plane (pushes with

a component to the right) because the lateral symmetry of our character allows us to mirror the controls to cover the remaining directions. The convex hulls of the controllers nest as shown in Figure 6.6. This suggests an integrated controller that can utilize multiple balance strategies by choosing an appropriate strategy based on the direction and magnitude of the observed push.

We order the strategies S_i by their balance recovery capabilities from low to high, i.e., in-place strategy(ankle and hip strategies coactivated), single stepping, and double stepping. The domain of a particular strategy $\Omega(S_i)$ is defined as $\Omega(S_i) = H(S_i) - H(S_{i-1})$. The sample points $C(S_i)$ used to define a strategy initially consists of the points on its domain boundary, i.e., the 10 points $C(S_i) = \mathbf{s}_{i_j} \cup \mathbf{s}_{i-1_j}$. We desire that the controls interpolated from $C(S_i)$ work for all points $\mathbf{s} \in \Omega(S_i)$. To verify this, we run the controllers on a densely-sampled grid of test perturbations within the bounding box of its domain. We denote a successful balance recovery with a '+', and a failure with a '×', as shown in the lower row of Figure 6.5. The middle column is the result for the single-step controller. As we can see, the small number of boundary point samples are already enough for the controller to work for the whole domain.

For the double stepping controller there exist regions where the initial small set of boundary point samples is insufficient to guarantee success over the full domain that they enclose. This is perhaps because the sample points are widely scattered across a large region and that the double-stepping motion is simply more complex in nature. We remedy this by adding more sample points in the interior of $C(S_i)$. The centroids of the failure region are chosen as points for placing additional samples. We do this iteratively until the controller works for the whole domain.

We emphasize that this integration method is independent of the different sub-controllers themselves, as long as each sub-controller defines a clear domain. For example, we can potentially replace our single-stepping controller with that reported in [Kudoh et al. 2006].

6.5 Results

Figure 6.6 illustrates the nested capabilities of the various types of balance strategies. Not included in this figure is the ability to do multistepping, i.e., to take multiple walking steps to dissipate the momentum acquired from the push. The shape of the controller domains has been characterized using a series of simulations in a set of 8 directions and recording the maximum push magnitude for which a given control strategy can succeed. These data points are illustrated and are used as the basis for constructing the control envelopes.

All controllers are tested with the simulation engine ODE (Open Dynamics Engine, http://ode.org/), on a full 3D human model with 30 internal DoFs. We refer readers to [Laszlo et al. 1996; Zhao 2004] for the detailed kinematic and dynamic parameters

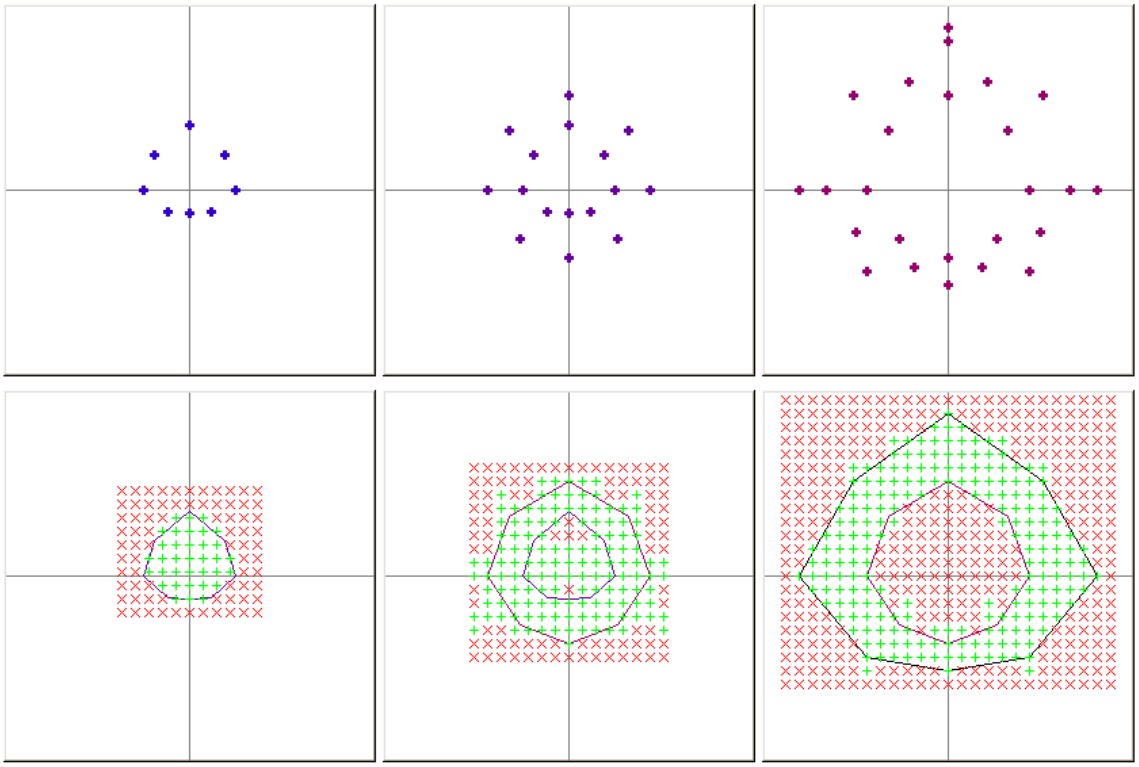


Figure 6.5: Left column: in-place strategy. Middle column: single stepping. Right column: double stepping. Upper row: the control points for each strategy. Lower row: Nested polygon outlines the domain of a controller. Green '+' marks the successful region of the controller. Red '×' marks the failure region. The interval between the marks are 50 N.

of this humanoid model. Before the perturbations are applied, the virtual character adopts a standard quiescent stance pose that has the feet placed at shoulder width. The pushes we apply are impact forces of 0.2 seconds at the chest level. During the pushes, we delay the activation of the stepping controllers until the end of the $0.2s$ push in order to mimic the latency of human sensory-motor feedback loops [Kawato 1999].

For all the controllers, the recoverable backward pushes are significantly less than those of the forward or sideways pushes. This conforms with our motion capture experiments with real humans, and we deem it to be a consequence of the anthropomorphic model. Also from the success of the relatively sparse control points for the stepping controllers, we conclude that our stepping controllers are smooth and robust, with respect to variations of push directions and magnitudes.

We now quantitatively compare our results to other results from the literature. [Wooten 1998] applied pushes below 150 N from different directions over a 0.25 second interval. In [Kudoh et al. 2002], 500 N forward push, 300 N backward push, and 200 N sideways push, each last 0.1 seconds are tested. The disturbance forces used in

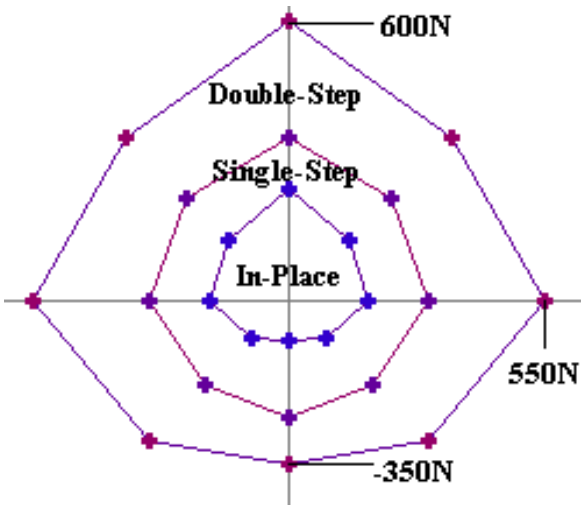


Figure 6.6: The domains of the various balance control strategies implemented in our system. The vertical and horizontal axes represent the forward (sagittal) and lateral magnitude of a 200ms push. The interior of the labeled regions represent the range of pushes that can be accommodated by each type of control strategy.

[Abdallah and Goswami 2005] is 300 N for 0.1 seconds. The perturbations reported in [Komura et al. 2005b] range from 0.1 $kg \cdot m/s$ to 0.2 $kg \cdot m/s$ in linear momentum[1], and 12.0 $kg \cdot m^2/s$ to 24.0 $kg \cdot m^2/s$ in angular momentum. The maximum push reported in [Kudoh et al. 2006] is 300 N forward for 0.4 seconds to the CoM.

A summary comparison of published results is shown in Table 6.1. We note that a performance comparison of balance controllers based solely on the magnitude of pushes may be misleading, because the directions, locations, and durations of the pushes may be different, and the underlying kinematic and dynamic models are usually different. Since from a static state, the linear momentum injected into a system is $\Delta P = f\Delta t = m\Delta v$, we believe that the maximum CoM velocity caused by the push, i.e., the momentum normalized by the total mass, is possibly a better choice for performance comparison. Quantitatively, for forward pushes with our controller the maximum recoverable CoM velocity is $1.27m/s$; for pushes sideways, the maximum velocity is $1.24m/s$; for backward pushes, it is $0.84m/s$.

We also tested the robustness of our controllers with respect to model variations, such as limb mass and limb length. We can decrease the mass of every limb up to 5% and the control for the single stepping works without any changes. If we decrease the mass by 10%, one out of the ten control points fails, and small adjustments (< 2 degrees computed automatically) are needed to make them work again. If we decrease the mass by 50%, the automatic control search algorithm fails to return meaningful control for two out of the ten control points. We also tried to increase the shank

[1]Judging from the magnitudes, there may be an error in the reported numbers.

paper	model	maximum pushes experimented		controller
		direction	ΔP $(kg \cdot m/s)$	
[Wooten 1998]	humanoid	forward	20	omnidirectional
		sideways	30	in-place only
		backward	16.25	
[Kudoh et al. 2002]	humanoid	forward	50	omnidirectional
		sideways	20	in-place only
		backward	30	
[Abdallah and Goswami 2005]	single leg plus HAT	forward	30	single direction in-place only
[Komura et al. 2005b]	humanoid	forward	0.2[1]	single direction during gait
[Kudoh et al. 2006]	humanoid	forward	120	single direction
ours	humanoid	forward	120	omnidirectional
		sideways	110	multiple strategies
		backward	70	

Table 6.1: Summary of work dealing with push recovery.

length, for which the unaltered control can support a 6% shank length increase.

Figure 6.7 shows initial and final foot configurations resulting from pushes of various magnitudes and directions. Figure 6.8 shows selected key poses from some of the simulated motions. For the complete motions, we refer the reader to the video http://www.cs.ubc.ca/∼kkyin/animation/Yin_SUB07.mov.

6.6 Discussion and Future Work

We have presented an omnidirectional, multi-strategy control framework for humanoid balance in response to unexpected external pushes. The stepping strategies rely on learning a model of where to step as a function of the CoM velocity resulting from pushes. The design of the stepping controllers can be informed by motion capture examples. The resulting balance controllers work in real-time to maintain the balance of a 30-DoF simulation of a humanoid. The domains of the various control strategies are characterized in detail, and we compare our results with a summary of other published push recovery strategies. We qualitatively compare our results with motion capture examples as well.

In the future, we wish to deal with a number of outstanding issues. The swing leg can collide with the stance leg during the course of balance recovery. This behavior can be predicted and corrected for. Additional strategies can be added, such as an arm rotation strategy. The demonstrated balancing skills should be integrated with models of other motor skills.

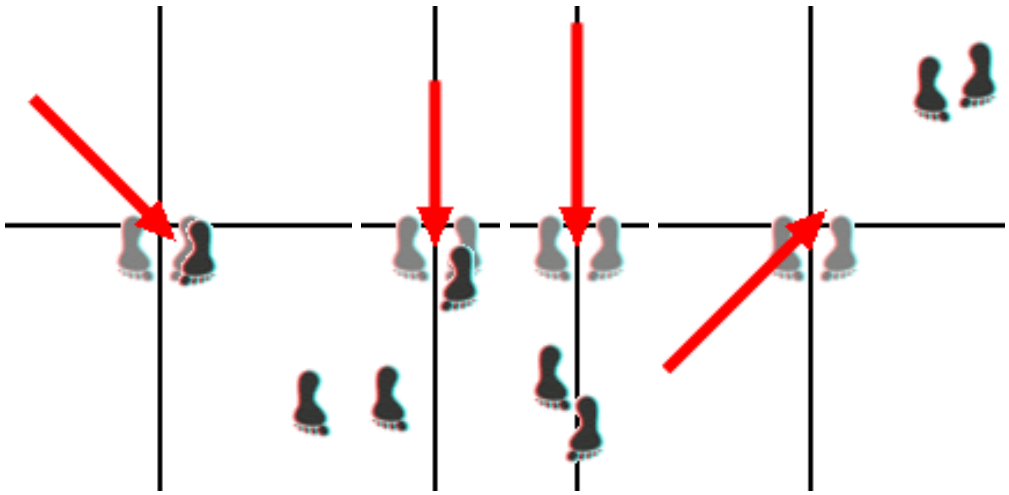


Figure 6.7: Initial (light grey) and final (dark grey) foot configurations for a variety of pushes (red arrows). The left two figures show single step responses. The right two figures show double step responses. In all cases shown the right leg steps first.

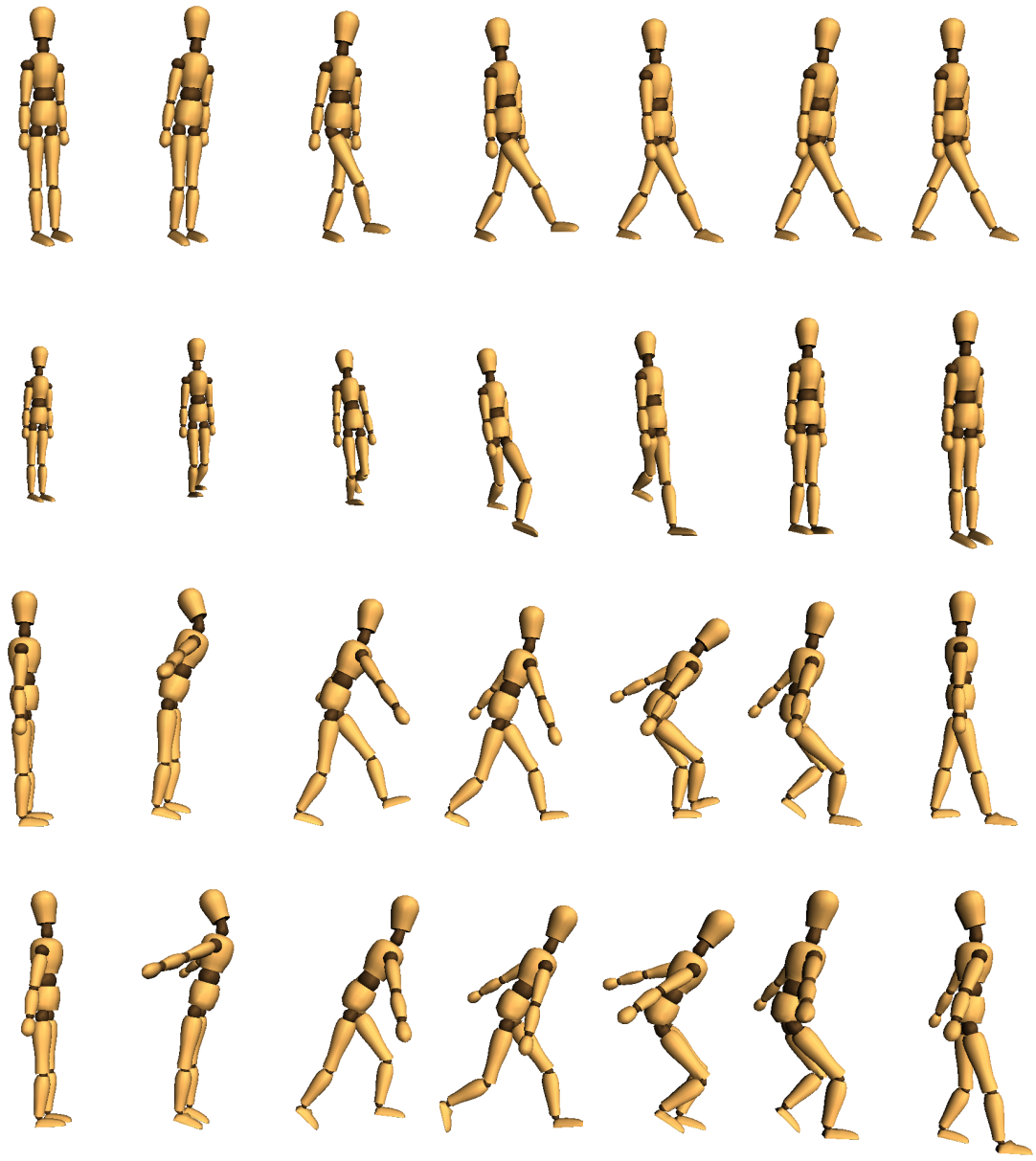


Figure 6.8: First row: single stepping upon a forward lateral push. Second row: double stepping upon a backward lateral push. Read from right to left. Third row: double stepping upon a forward push. Last row: the motion capture data used to construct the controller for the response in the third row.

Chapter 7

Dynamic Animation of Locomotion: SIMBICON

The previous chapter dealt with pushes to standing characters, and the goal was to stop the movement of the body and to rapidly recover balance. This chapter looks at the more general problem of walking and running with different gaits and styles, while being robust to pushes and terrain changes.

Physics-based simulation and control of biped locomotion is difficult because bipeds are unstable, underactuated, high-dimensional dynamical systems. We develop a simple control strategy that can be used to generate a large variety of gaits and styles in real-time, including walking in all directions (forwards, backwards, sideways, turning), running, skipping, and hopping. Controllers can be authored using a small number of parameters, or their construction can be informed by motion capture data. The controllers are applied to 2D and 3D physically-simulated character models. Their robustness is demonstrated with respect to pushes in all directions, unexpected steps and slopes, and unexpected variations in kinematic and dynamic parameters. Direct transitions between controllers are demonstrated as well as parameterized control of changes in direction and speed. Feedback-error learning is applied to learn predictive torque models, which allows for the low-gain control that typifies many natural motions as well as producing smoother simulated motion.

Section 7.1 provides an overview of approaches, including our proposed approach. Closely related work is visited in Section 7.2. Section 7.3 describes our balance control strategy. Section 7.4 and 7.5 describes the controller design process, either manually or from motion capture examples. Detailed experimental results and discussions are given in Section 7.7. We conclude the chapter with a discussion of future work in Section 7.8.

7.1 Overview

Locomotion is at the heart of many motions, both real and animated. Animated motion is most often created directly by animators using keyframing, or by capturing and then processing human motion. However, these approaches fail to scale to the very large set of possible motions that might arise in a realistic environment. For example, there are an infinite number of ways in which two characters might bump into each other or in which a character may move through a constrained, unpredictable environment. Algorithmic approaches have the potential to be more general and capable of generating families of motions rather than individual motions.

A subset of algorithmic approaches take physics into account. These include trajectory optimization methods, or alternatively, developing controllers to drive forward dynamics simulations. Controller-based approaches have the advantage that they can synthesize motion at interactive rates and produce motion by using feedback strategies to continually adapt to the real-world as necessary.

The control of walking and other biped locomotion gaits has been of long-standing interest to the robotics and computer graphics communities. It is a challenging problem for many reasons. Walking bipeds are unstable and underactuated, and their control involves high-dimensional states and high-dimensional actions. Locomotion involves joint limit constraints, torque-limit constraints, contact constraints, and contact impacts. Locomotion may have a number of contradictory goals, including robustness and energy usage. Lastly, while data-driven approaches have been very successful at generating kinematic models of locomotion, it is unclear whether such strategies can be successfully adopted to learn control strategies for dynamic simulations.

There exists a vast literature related to the control of bipedal walking, much of it in the robotics, control, and biomechanics communities. Common approaches to locomotion control include: (a) the use of passive walking as a starting point for the design of active walkers; (b) the use of "zero moment point" control; (c) using a fixed control architecture and applying parameter search to find the parameter settings that yield successful walking gaits; and (d) developing feedback laws based upon insights into balance and locomotion. Our proposed framework, named SIMBICON[1], builds on the last of these approaches.

7.1.1 Overview of Our Approach

The starting point is the use of a simple finite state machine or *pose control graph*, the same as in the last chapter. Each state consists of a body pose representing target angles with respect to their parent links for all joints. All individual joints attempt to drive towards their target angles using proportional-derivative (PD) controllers.

[1]SIMBICON is an acronym for SIMple BIped CONtrol.

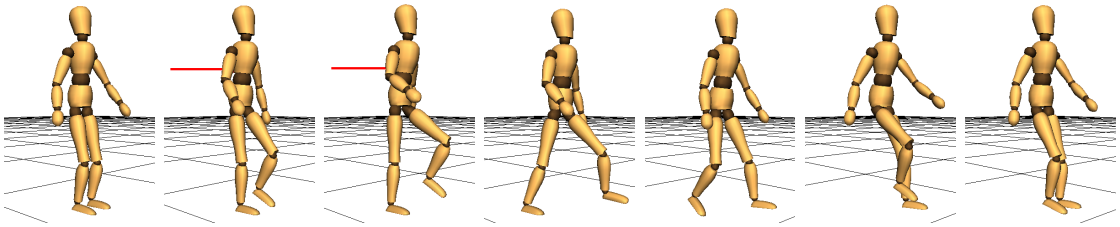


Figure 7.1: An example result. A walk controller reconstructed from motion capture data responds to a 350N, 0.2s diagonal push to the torso.

Transitions between states occur after fixed durations of time, or after a new foot contact has been established. Walking gaits can be modeled using as few as four states, while running gaits can be modeled using as few as two states.

The pose control graph as described thus far does not have any notion of balance and thus does not produce robust locomotion. However, a small set of modifications to this basic design does result in robust locomotion. First, we require that both the torso and the swing-leg femur (i.e., swing hip) have target angles that are expressed with respect to the *world* frame, unlike the remaining links which have target joint angles expressed with respect to their parent links. In order to make the resulting torques be physically realizable without the use of external torques, the stance-hip torque is left as a free variable. Second, a feedback term is added to continuously modify the swing hip target angle as a linear function of the center of mass (CoM) position and velocity. This provides a robust balancing behavior by changing the future point of support. In contrast, we only adjust controls once per cycle for the limit cycle walking controller in the last chapter.

We can mimic the style of motion capture data by replacing the individual control states with tracking of a desired motion, using the same mix of local-and-world coordinate tracking. While such tracking normally requires high gains and a resulting stiff and reactive motion, we can apply feedback error learning (FEL) in order to produce a control solution that relies largely on predictive feed-forward torques. As a result, the final motion requires only low-gain feedback and exhibits considerably less unnatural oscillation because of the anticipatory nature of the predictive torques.

The resulting controllers can be applied to 2D and 3D bipeds of human proportion and mass distributions to produce many different styles of locomotion using physics-based forward dynamics simulations. Only physically-valid internal torques are used to produce the motion, and thus the approach may also extend to humanoid robots. Individual controllers are robust to large pushes and significant terrain variation. Controllers can be interpolated and parameterized. Direct transitions between many of the controllers are demonstrated. An example result is shown in Figure 7.1, where a walking controller constructed from a motion capture example recovers from a large push.

7.2 Related Work

The wealth of literature on walking control precludes an exhaustive review of the literature. Our discussion aims to touch on the major categories of techniques, as well as focussing on the specific work that is closest to our own.

The seminal work of Raibert, Hodgins, and colleagues [Raibert 1986; Raibert and Hodgins 1991; Hodgins 1991; Hodgins et al. 1995] contains key insights into producing robust hopping and running gaits. At the heart of this research is a three-way decomposition of control of hopping height, control of torso pitch, and control of hopping speed. Swing foot placement provides the basic mechanism for controlling balance from stride to stride. The algorithms are applied to the control of running for biped robots in [Raibert 1986; Raibert and Hodgins 1991; Hodgins 1991], walking for biped robots in [Hodgins 1991], and running for human characters in [Hodgins et al. 1995; Hodgins and Pollard 1997]. We are unaware of demonstrations of the algorithm being used to control walking for virtual humans. We note that the control of swing foot placement is also found in other walking algorithms such as [Miura and Shimoyama 1984; Laszlo et al. 1996; Kuo 1999].

Our control framework integrates a number of the ideas from previous work, while being identical to none. We differ in several respects from the Raibert-style hopping control. The balance feedback mechanism we propose makes *continuous adaptations* during a locomotion cycle, and uses both the *position and velocity* of the center of mass. This latter information helps establish the current phase of an ongoing step, and thus it is more informative than the velocity alone. [Hodgins 1991; Raibert and Hodgins 1991] use only the velocity for control of hopping, sampled once per hop, and use of a fixed step length for walking. We note that [Miura and Shimoyama 1984] is an example of an inverted pendulum control strategy that employs continuous feedback based on the inverted-pendulum body angular position and angular velocity. The framework we propose is simpler in many respects and will be demonstrated to be capable of producing a significantly larger variety of gaits.

A widely-studied class of control algorithm can be developed by computing and then tracking trajectories that are known to be physically feasible and therefore satisfy the zero-moment-point (ZMP) constraint [Vukobratovic and Juricic 1969]. Small disturbances to the motion can be accomodated by adjusting the ZMP dynamically during the motion. This approach has been shown to work well for the walking control of real robots such as the Honda Asimo P3 [Kaneko et al. 2002; Honda Motor Co. 2006; Kim et al. 2006]. Target trajectories can be derived through an optimization process, often informed by motion capture data [Dasgupta and Nakamura 1999; Popovic and Witkin 1999]. ZMP approaches need to include swing-foot placement if they are to deal with large disturbances. [Pai 1990; Pai 1991] present an interesting approach to specify the control of human-like walking. They do not specify walking in terms of desired trajectories but rather more weakly as a collection of assertions to hold.

Control can also be achieved by defining a parameterized control policy and then searching for parameter values that yield appropriate control behaviors. Our stepping controller in Chapter 6 belongs to this category. Searching directly in the often high-dimensional parameter space is known as *policy search*, and this has been applied with some success [Taga et al. 1991; van de Panne and Fiume 1993; Auslander et al. 1995; van de Panne et al. 1994; Tedrake et al. 2004; Sharon and van de Panne 2005]. To date, these techniques have not been able to demonstrate the scalability and robustness needed to make it a useful, widely applicable locomotion control technique for animation and robotics.

Reinforcement learning (RL) offers a long-term promise of being able to learn control strategies in a principled way. It has been applied in a number of ways to the control of walking [Tedrake et al. 2004; Nakanishi et al. 2003; Morimoto et al. 2004; Smith 1998; Morimoto and Atkeson 2003; Morimoto et al. 2005]. The high-dimensionality of the state spaces involved in the control of locomotion remains problematic, however, as does the need to design an appropriate reward function. The solutions do not exhibit the compactness and transparency which would afford control to animators in shaping the results. Many of the control strategies sample the state only once per step when computing a control decision, similar to using a Poincaré map [Morimoto et al. 2005]. Work in this area usually works with simplified models instead of humanoids, with [Smith 1998] being a notable exception.

Lastly, there is a body of work that uses special strategies that forego some of the physical fidelity in order to achieve desired plausible motions. One choice is to allow the addition of external forces as in [Wrotek et al. 2006]; another is to blend kinematic and dynamic motions resulting in a hybrid system [Zordan et al. 2005]. Other commercial systems use undocumented strategies, such as [NaturalMotion]. SIMBICON produces balanced locomotion behaviors directly with a forward dynamics simulation, and thus avoids the complications that mixed kinematic/dynamic solutions use.

The integration of a limited number of carefully crafted control strategies is demonstrated in [Wooten 1998; Faloutsos et al. 2001]. The interesting recent work of [Sok et al. 2007] parallels many of our goals and makes use of an optimization process to produce controllers for planar articulated characters that are capable of imitating motion capture data.

7.3 Balance Control Strategy

The control strategy can be described in terms of three elements: a pose-control graph, torso and swing-hip control, and balance feedback. Each of these elements is now described in further detail.

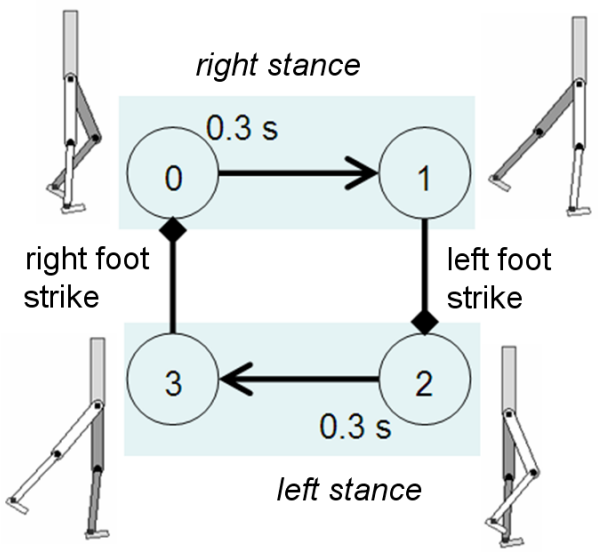


Figure 7.2: Finite state machine for walking.

7.3.1 Finite State Machine

Our controllers are based on finite state machines, with each state having its own target pose for internal joint angles, as shown in Figure 7.2. For symmetric gaits, pairs of states will be left-right symmetric, e.g., states 0 and 2, 1 and 3. Transitions between states happen after an elapsed time, e.g., state transition $0 \rightarrow 1$, or after foot contact, e.g., $1 \rightarrow 2$. If a foot contact has already occurred before entering a state having an outbound foot-contact transition, then the controller simply spends no time in that state. In any given state, the joints apply torques computed by proportional-derivative control, $\tau = k_p(\theta_d - \theta) - k_d\dot{\theta}$, in order to drive each joint to its desired local angle. The poses represent a desired set of joint angles and are typically not actually achieved. For example, while in state 1 in Figure 7.2, the biped's pose in practice has its swing leg extended forwards. However, its target state has the swing leg extended backwards and thus has a net effect of moving the swing leg backwards and down, bringing it into contact with the ground.

7.3.2 Torso and Swing-hip Control

The stance hip and swing hip are handled separately, as illustrated in Figure 7.3(a). First, there is a need to control the orientation of the torso with respect to the world frame. This can be accomplished using a virtual PD controller that operates in the world frame to compute a net torso torque τ_{torso}, as shown in the figure. Second, there is also a need to decouple swing foot positioning from the current torso pitch angle. This is accomplished through controlling the swing hip with respect to the world coordinate frame. The swing hip torque, τ_B, is thus also computed using a virtual

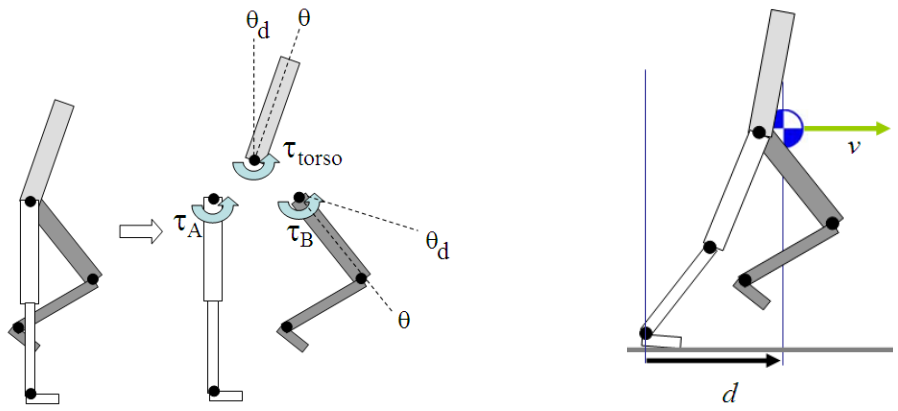


Figure 7.3: Elements of the balance control strategy: (a) Relationship between torso, stance-hip, and swing-hip torques; (b) Center-of-mass position and velocity.

PD controller that operates in the world frame. Last, there is a requirement that the virtual torques τ_{torso} and τ_B be realisable using only internal torques. We require that the desired value of τ_{torso} is in fact the net torque seen by the torso, $-\tau_A - \tau_B$. This is accomplished by computing the stance hip torque as $\tau_A = -\tau_{torso} - \tau_B$.

7.3.3 Balance Feedback

The last component of the control strategy is to apply a balance feedback strategy to the swing foot placement. We employ a feedback law of the form

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

to the swing hip, where θ_d is the target angle used for PD control at any point in time, θ_{d0} is the default fixed target angle as described in the FSM, d is the horizontal distance from the stance ankle to the center of mass (CoM) as shown in Figure 7.3(b), and v is the velocity of the center of mass. The midpoint of the hips can be used as a simple and effective proxy for estimating the position and velocity of the center of mass. We use this simplification in both 2D and 3D.

The feedback gain parameter c_d is important for providing balance during low-speed gaits or in-place stepping. Consider a situation for an in-place (desired zero velocity) walking gait with current velocity $v = 0$, and two possible CoM positions $d_a = +10cm$, $d_b = -10cm$. In the first case, there is a need to step forward quickly, while in the second case there is a need to step backwards quickly in order to recover balance. The combination of (d, v) provides complete information about the current position in the gait cycle, i.e., the current phase, whereas v alone only provides information with regards to the current velocity error.

In order to extend the control scheme to 3D, the control strategy is applied in both the sagittal and coronal planes. Balance feedback in the coronal plane uses an analogous measure of d, v in order to make alterations to the lateral placement of the swing foot using the swing hip.

The balance feedback can be extended more generally to multiple joints using the form

$$\theta_{\mathbf{d}} = \theta_{\mathbf{d0}} + \mathbf{F} \begin{bmatrix} d \\ v \end{bmatrix} \tag{7.1}$$

where $\mathbf{F}$ is an $n \times 2$ matrix with feedback coefficients to the desired target joints. We use this more general structure to add stance ankle feedback for quiescent stance poses, for example.

In the two following sections, we provide details on how the control parameters are set and the creation of controllers that imitate locomotion styles from motion capture data.

7.4 Manual Controller Design

Given the controller architecture described in the previous section, we need methods for choosing the number of states and the parameters of each state. The resulting parameters should satisfy the requirements of the animator or control-system designer. Unfortunately, it is difficult to precisely pin down such requirements. Criteria for locomotion may include measures of style, robustness to perturbation, and energetic efficiency, all of which may push the solution in different directions and with design compromises that will be unknown in advance. Therefore, before resorting to more complex schemes, we first investigate manual interactive design of the required finite state machine.

The control parameters can be grouped into several categories: (a) number of states and state-transition parameters; (b) the balance feedback gains, c_d and c_v; (c) the target poses for each state; (d) the initial state for using the controller; and (e) the joint limits, torque limits, and PD-controller gains. In our work, we fix the parameters belonging to category (e) and document these in the results section. The remainder of our discussion focuses on the other parameters.

The choice of the number of states reflects the detail with which to model the various phases of a locomotion gait. We use four states to model our walking gaits, consisting of two symmetric walking steps. Each step has two states, the first of which lifts the swing foot upwards and forwards for a fixed duration of time, and the second of which drives the swing foot towards the ground until contact is made. This model is capable of many different walking styles, both forwards and backwards. The FSM states also serve as a coarse model of the phase of the motion when switching

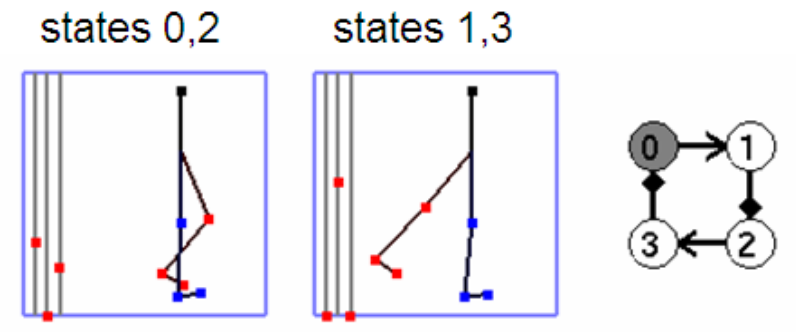


Figure 7.4: Graphical interface for adjusting controller parameters. Sliders on the left control Δt, c_d, and c_v.

between controllers. Thus, if a request is made to switch from one walk style to another, this is done by transitioning from state n of one controller to state $n+1$ of another controller. For this reason, while our running gaits can be modeled using simple two-state controllers (one for each running step), we add two zero-duration dummy states in order to have the same four-state structure as for the walking gaits. This allows for transitions between walking and running gaits. Our skipping controller will have 8 states, reflecting its more complex sequence of actions.

We begin controller design using the planar biped model, and then use the resulting parameters as a starting point for the design of corresponding 3D controllers. We use a graphical user interface (GUI) to allow a user to directly explore the parameters settings associated with each of the controllers states, as shown in Figure 7.4. Users can immediately observe the effect of parameter changes reflected in an ongoing simulation. Three sliders on the left of each state GUI are used to set the state duration, c_d, and c_v parameters. The target pose parameters are set by using the handle points on the stick figure. The target poses for the torso and the swing femur are interpreted with respect to the world frame. The target pose angle for the stance femur is ignored by the controller, given that the stance hip torque is treated as a free parameter whose value is determined from the torso and swing-hip torques. All the remaining joint angles define target angles with respect to their parent's coordinate frame. The interface readily exposes the key-frame like nature of many of the controller parameters.

The most important parameters for each state are the state duration Δt and the target angles for the swing hip and swing knee. Because the resulting motion style is most heavily dependent on only these three parameters per state, it becomes relatively easy for users to interactively explore their settings to yield desired motions. The ankles make a significant contribution to some styles, such as the skipping gait. The stance knee is usually almost straight. The torso is usually desired to be vertical. The balance feedback gains are set in a similar fashion across many of our controllers.

The design of a stepping-in-place gait for 2D locomotion represents a good starting point that can then be modified for the design of other motions. The target angles, as shown in Table 1, look very much like a simple pair of keyframes, one in a standing posture, and another with the swing leg in the air in a bent pose as one might expect for stepping-in-place. This leaves very few remaining parameters to set, principally the duration of the leg lift pose and the balance feedback gains for the swing hip.

Small changes ($\pm 15^{\circ}$) to the desired torso pitch can be easily accomodated by treating the extra torque produced by gravity as a disturbance. During locomotion, the torso may exhibit a somewhat unnatural bobbing motion. This is the result of the torso servo always reacting to the motion of the hip, rather than anticipating it. We address this in the section on feedback-error learning.

A reasonable choice of initial state is required in order for a controller to function as designed. In practice, the balance feedback terms endow the controllers with relatively large basins of attraction, as demonstrated by their robustness to external pushes and changes in terrain, and the ability to transition directly between many of the controllers. We begin our walking controllers from a double stance state with a moderate forward velocity ($1m/s$), although we note that our basic forward walking controller can begin just as well from rest. We note that symmetric controllers can exhibit asymmetric gaits from some initial states while producing symmetric gaits from other initial states. This difficulty can be overcome by using initial states closer to the desired limit cycle.

7.5 Controllers from Motion Capture Data

An alternative to manual design is to use motion capture data as the basis for developing a controller. This allows a kinematic motion to be imported into a dynamic setting. Whereas kinematic motion capture data cannot be made to stumble for an unseen step or respond to a push, a style-mimicing controller allows for these effects.

We develop style-mimicing controllers as follows. Given 3–7 cycles of motion capture walking data, we apply Fourier analysis to a representative joint, such as the right hip, in order to extract the period T of the walking cycle, which is estimated using the primary frequency ω of the Fourier transformation. We then select the next several largest Fourier coefficients corresponding to frequencies that are an integer multiple of ω. When the reconstruction error reaches a preselected accuracy, we discard the remaining Fourier components. This filters the original motion capture data to a smooth periodic motion Θ that reflects an averaged walk cycle of the given style and represents it using a compact set of coefficients.

The extremal points of Θ are detected automatically. The time t_m of the largest right hip flex is assigned a fixed phase $\phi(t_m) = 0.25$ in the walk cycle, where $\phi \in [0, 1]$. States 1 and 3 are discarded from the FSM in Figure 7.2. The remaining two

states serve the purpose of differentiating between the left-stance and the right-stance phases. The transition between the states occurs on foot contact, which is expected to occur at approximately $\phi = 0.5$ and $\phi = 1$. The double-stance phase is considered to be part of the stance phase that has just begun.

The trajectory Θ serves as a target trajectory in place of the target poses used in the manually-designed controllers. Within each state, the joint angles individually track the motion capture trajectories $\theta_{d0} = \Theta(\phi(t, T))$ using PD-controllers. The phase is reset to 0 or 0.5 upon transitioning to the next state. As is the case for the manually-designed controllers, the torso angle and swing-hip angle do their tracking with respect to the world frame. Similarly, the stance hip does not track its motion-captured counterpart and, as before, its torque is computed from the known torso and swing-hip torques. The PD-controller, however, is changed from $\tau = k_p(\theta_d - \theta) - k_d\dot{\theta}$ to $\tau = k_p(\theta_d - \theta) - k_d(\dot{\theta} - \dot{\theta_d})$. The $\dot{\theta_d}$ term helps in tracking the target trajectory with minimal time lag.

Controllers based on motion capture data apply balance feedback to both the swing hip and, for slow walks, to the stance ankle using Equation 7.1. While a fairly broad range in values result in stable gaits, we currently tune these by hand in order to yield a robust gait and a strongly-attracting limit cycle which will be required for the successful application of feedback error learning, as will be discussed later.

The controller will not perfectly imitate the motion capture reference motion for a number of reasons. First, the original motion capture data may contain noise from data capture and data processing and may not be dynamically consistent. Second, the physical system parameters of the simulated human may not exactly match that of the motion-capture actor. This includes link dimensions, joint placement, mass and inertial parameters, and joint gains. Additionally, we forego tracking of the stance hip angles in order to insert the balance-feedback mechanism. The resulting motion is thus encouraged to imitate the overall style of the reference motion, but does not precisely match other parameters that could also be used to characterize gaits, such as step-length or velocity. Lastly, in order to closely follow the reference motions, the tracking control requires high-gain PD controllers. These can be lowered using the feedback-error learning scheme discussed in the following section.

7.6 Feedback Error Learning

The controllers described thus far produce motions that are quite robust to disturbances and are able to closely track reference trajectories. However, the high gains used in the feedback loops are representative of stiff movements. Also, the torso pitch angle may oscillate about its desired position because it is always reacting to the movement of the hip rather than anticipating it.

Human motor control commonly increases the mechanical impedance of the control

as a strategy for producing robust motions in environments where perturbations are expected, and uses low-impedance control when the environment is highly predictable [Takahashi et al. 2001]. The low-impedance control in predictable environments can be achieved by using anticipatory "feed forward" torques together with low-gain feedback. The latter is still necessary to deal with small deviations from the desired trajectory that may always be expected. The feed-forward torques are also thought to be necessary in order to deal with the slowness of the human nervous system, as compared to the fast response that would be required for purely feedback-based control (Section 2.3.2).

Previous work using tracking of upper-body motion capture trajectories requires high-gains in order to track well [Zordan and Hodgins 2002]. At the instant of an unexpected impact, the gain is lowered for a short duration in order to mimic the low-impedance control normally exhibited during skilled motion, before being increased again to resume tracking. The ephemeral low-gain portion of the motion is implausible in that it would not track the default motion in the absence of the disturbance. In Chapter 5 we use inverse dynamics in order to estimate the feed-forward torques that would normally be in effect during skilled motion, although we do not deal with issues of balance there. In contrast, in this chapter we apply feedback error learning (FEL) in order to learn feed-forward torques, which then allow our controllers to operate with low tracking gains. Because the low-gain motions are less robust than the high-gain motions, we can optionally *increase the gains* for some time after impact after a reaction-time delay (150ms) in order to simulate a natural perturbation-recovery reaction.

Feedback error learning is a form of adaptive control and allows for the learning of the inverse dynamics of a system in order to reproduce given motion trajectories [Kawato et al. 1987; Nakanishi and Schaal 2004]. In its most general form, the feed-forward function learns $\tau = f(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})$, where $\mathbf{x}, \dot{\mathbf{x}}$ is the current system state (positions and velocities) and $\ddot{\mathbf{x}}$ is the vector of commanded accelerations. We move away from this general form and instead learn the feed-forward torques as a function of the current phase of the motion: $\tau = f(\phi)$. We estimate ϕ using $\phi = t/\hat{T}$, where t is the current elapsed time in a given state and $\hat{T}$ is the current estimate of the period. For repetitive motions such as a walk cycle, these simplifications work well. To our knowledge, FEL has not been successfully applied to a dynamical system as complex as a fully simulated virtual character capable of a variety of robust locomotion behaviors.

Our implementation of FEL divides the phase ϕ uniformly into N bins. We use $N = 20 - 1000$. The current phase bin is given by $n = tN/\hat{T}$. Each bin uses a low-pass filter of the form

$$v'_{ff} = (1 - \alpha)v_{ff} + \alpha(v_{ff} + v_{fb}) \tag{7.2}$$

to update the feed-forward torques. Here v_{ff} and v_{fb} represents the feed-forward and

feed-back torques applied for that phase of the motion corresponding to bin n. We use a learning rate of $\alpha = 0.1$. The feed-forward values are initialized to zero. During the learning process (and afterwards), it is the sum of feed-forward and feedback torques that is applied, i.e., $v_{ff} + v_{fb}$. FEL can be applied to one joint at a time, or to all joints simultaneously.

High learning rates may yield to convergence problems because the use of feed-forward torques will influence the resulting motions. While it is difficult to obtain analytic guarantees of convergence of FEL for complex dynamical systems, we have not experienced any convergence problems with our chosen learning rate. It is also useful to limit the magnitude of the feed-forward torques because some predictable disturbances can never be fully accomodated. An example of this is the force impulse caused by foot contact, which instantaneously creates a small change in angular velocity for the torso and requires an equivalently instantaneous control impulse in order to fully ensure that the torso never exhibits any pitch.

7.7 Results

We apply the SIMBICON framework to simulated 2D and 3D bipeds having human-like proportions and mass distributions. Figure 7.5 shows the models and their degrees of freedom.

The 7-link planar biped has a 70 kg trunk, 5 kg upper legs, 4 kg lower legs, and 1 kg feet. The respective largest dimensions are 48 cm, 45 cm, 45 cm, and 20 cm. A combined head-arms-trunk (HAT) model is used, as is common in the walking simulation literature. The 2D biped is simulated using an optimized version of the Newton-Euler equations of motion. A spring-and-damper penalty-force ground contact model is applied to points at the front and back of the feet. PD gain values of $k_p = 300\ Nm/rad, k_d = 30\ Nms/rad$ are used for all joints. Joint limits are enforced on the hips and knees. Ground stiffness parameters are $k_p = 100000\ N/m, k_d = 6000\ Ns/m$. We use a Coulomb friction model with a friction coefficient of 0.65. A time step of 0.0001 s is used. We use torque limits of $1000Nm$, which can be lowered to 370 Nm for all motions except the fast run. The basic walk controller supports a torque limit of 90 Nm, below which it becomes weak-kneed and falls. With control, the simulation runs 5 times faster than real-time unoptimized on a 1.8 Ghz CoreDuo PC.

The parameters for the 3D biped model are the same as used in [Laszlo et al. 1996]. We scale the limb lengths to match our motion capture subject. The 3D biped is simulated using Open Dynamics Engine [ODE] version 0.6. Our simulation time step is 0.005 s. Contacts are modeled using constraints and an approximated Coulomb friction cone, solved as a linear complementarity problem (LCP). We use a friction coefficient of 0.8, which is typical for a rubber sole. The coefficient of

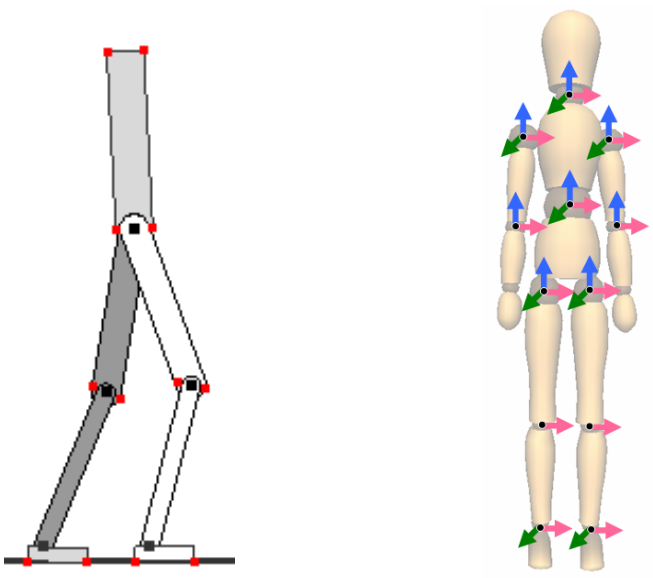


Figure 7.5: Degrees of Freedom (DoF) of models. Left: 2D model with 6 internal DoFs, 9 DoF in total. Right: 3D model with 28 internal DoFs, 34 DoF in total.

restitution for collisions is assumed to be zero. Torque magnitudes are limited to k_p The largest k_p value is 1000, used for the waist joints. All other joints use $k_p = 300$ or less. We use $k_d = 0.1k_p$. With control, the unoptimized simulation runs 1.2 times faster than real time on a 3.2 Ghz Pentium 4.

7.7.1 2D Biped Locomotion

A set of 12 periodic gaits have been authored using the GUI described in Section 7.4. The parameters for these controllers are given in Table 7.1. We designed these gaits to achieve a wide variety of motion styles using a small number of states. They have not been designed to be optimal gaits with respect to any given criterion. We have also authored acyclic controllers for stopping and remaining balanced on two feet, stopping and remaining balanced on one foot, and taking a single large step in the middle of a longer walking sequence. A subset of the motions are illustrated in Figure 7.6.

The running controllers do not have a strong preference to run at a particular speed. As such, they can be 'pushed' to run at various speeds. Parameterized control of speed is likely feasible, although was not investigated. As compared to [Raibert and Hodgins 1991], our framework uses no explicitly-computed injection of energy to maintain a given flight time during running or skipping. The control laws are identical for all FSM states, modulo the change of legs fulfilling stance-leg and swing-leg roles, and are governed by the target angles and feedback gains. This supports a style of running that, qualitatively speaking, looks less like hopping than previous

work [Hodgins et al. 1995; Hodgins and Pollard 1997].

Controller Transitions

Controllers are bound to keystrokes and it is possible to interactively request transitions between the controllers. This is accomplished during state transitions, jumping from state n of controller A to state $n+1$ of controller B. An example of the kind of transitions that be successfully executed is as follows: *walk* → *in-place walk* → *stand* → *back-flip* → *in-place walk* → *walk* → *big-step* → *walk* → *in-place walk* → *high-step walk* → *in-place walk* → *scissor hop* → *walk* → *skip* → *walk* → *bent walk* → *walk* → *crouch walk* → *walk* → *in-place walk* → *backwards walk* → *in-place walk* → *back-and-forth stepping* → *walk* → *run* → *fast run*. A portion of this is shown in the video http://www.cs.ubc.ca/∼kkyin/animation/Yin_SIG07.mov.

In the absence of specially-designed transition motions, not all controller transitions are feasible. For two controllers whose motions are significantly different, the basins of attraction for each may not overlap as needed for direct transitions. Direct transitions are possible between many of the walking and running gaits. However, some gaits are significantly more sensitive to the required initial state, such as the backwards walk which first requires transitioning to the in-place walk. The "scissor hop" is particularly sensitive to its initial state. Its basin of attraction does not fully overlap the limit cycle of the in-place walk, and thus requires being in a particular phase of its walk cycle in order to ensure a successful entry transition. The controllers for standing on one foot also has a very limited basin of attraction due to its need to balance without stepping.

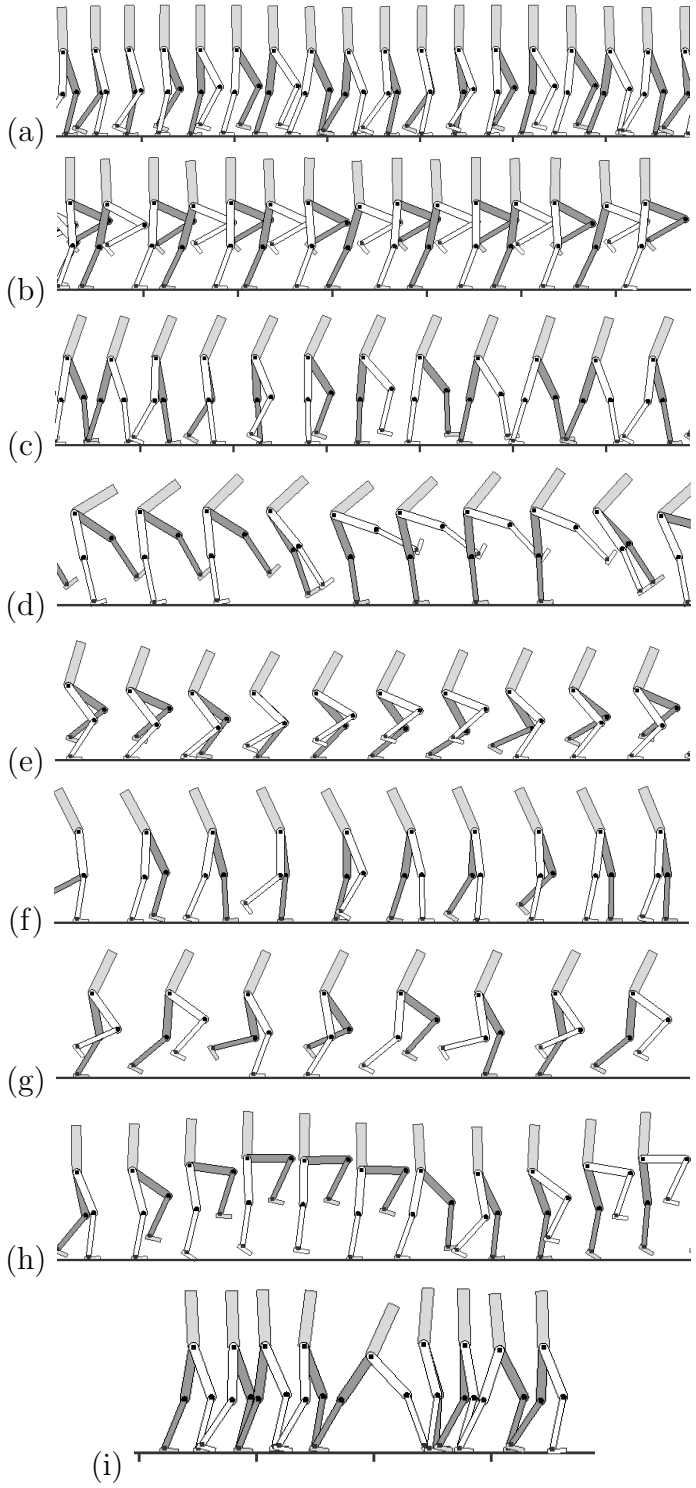


Figure 7.6: A subset of the manually-designed controllers for the planar biped. (a) walk; (b) high-step walk; (c) bent walk; (d) scissor hop; (e) crouch walk; (f) backwards walk, right-to-left; (g) fast run; (h) skipping; (i) big step.

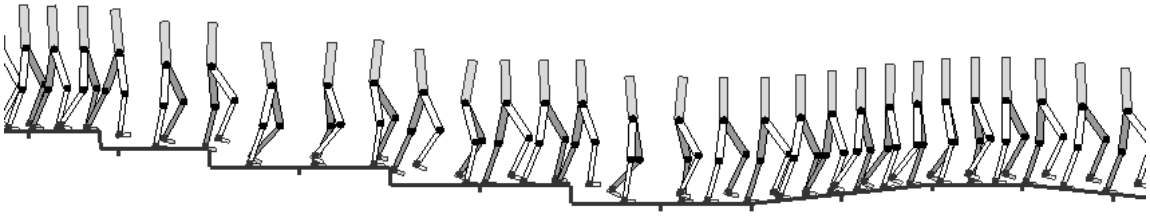


Figure 7.7: A single controller for a planar biped responds to unanticipated changes in terrain.

Robustness

The robustness of the walk controller to variations in terrain is shown in Figure 7.7. The terrain includes unanticipated downward steps of $20cm$ and slopes of ± 6 degrees. The robustness of the gaits with respect to unanticipated pushes was tested by applying 10 pushes at $5s$ intervals, which serves to sample various phases of the gait while also allowing the biped time to fully recover between pushes. Any single stumble from which the biped cannot recover is deemed a failure. The walking controller can withstand $0.1s$ duration pushes of up to $600N$ forwards and $500N$ backwards at all 10 sampled points in the locomotion cycle. Other gaits are more sensitive to disturbances. For example, the skipping gait can withstand $0.1s$ duration pushes of up to $40N$ forwards and $50N$ backwards. Larger pushes, as measured by their induced change in momentum, $F\Delta t$, can be sustained by increasing Δt and decreasing F. Specific portions of the walk cycle can also withstand larger pushes.

Feedback Error Learning

The application of feedback error learning to the 2D biped torso decreases its oscillation amplitude from 5° to 0.5° for a walking gait and unchanged feedback gains. Figure 7.8 illustrates the learned torque and the decreased amplitude of the feedback torques. Footstrike events occur at time $t = 0$ and $t = 0.36$ on the graph and cannot be fully anticipated due to their impulsive nature.

7.7.2 3D Biped Locomotion

Manual Controller Design

Controllers for 3D walking require twice the number of parameters as for 2D control because of the need for lateral (coronal-plane) control. Manually-designed controllers have been developed for two-foot hopping, three styles of forward walking, walking up a slope of 20 degrees, two styles of forward running, a sideways-walk Tai Chi movement, and skipping. Figure 7.9 illustrates the Tai Chi "cloud hands". Direct transitions are possible between most of the walking and running gaits. Many of our 2D control strategies work directly when applied to the 3D model and applying a

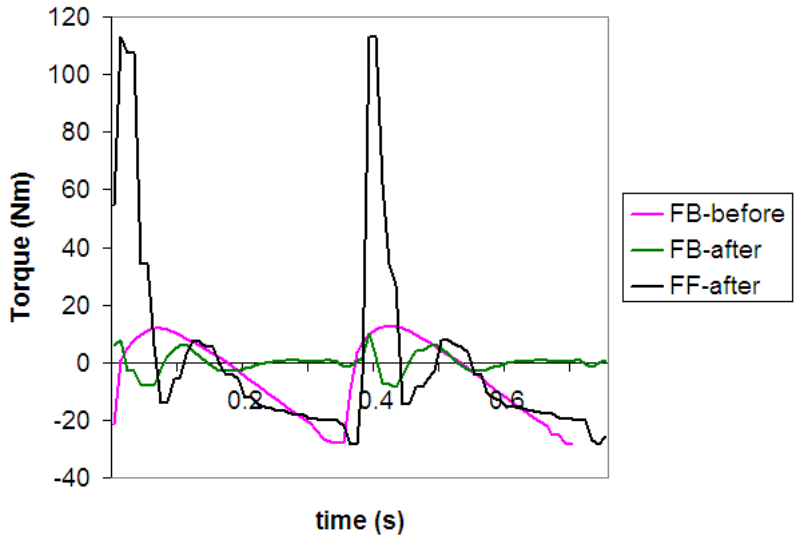


Figure 7.8: Feedback error learning (FEL) applied to the 2D biped torso. Illustrated are the feedback torques, before and after FEL, and the learned feed-forward torque.

common set of feedback gains for the lateral hip movement that is responsible for lateral balance. There is some variation in the style of the 2D and 3D motions, likely because our 2D and 3D models have different masses and proportions. For the few cases where the difference in models is problematic, minor adjustments to target angles and gains are sufficient to achieve a functional 3D motion.

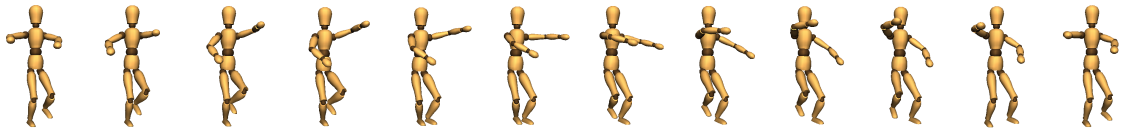
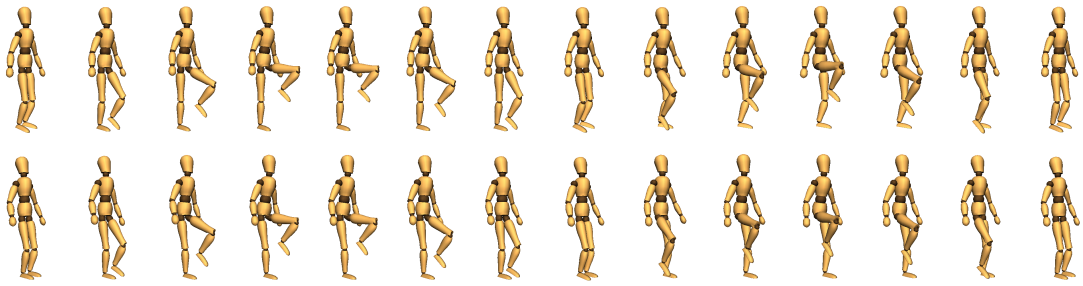


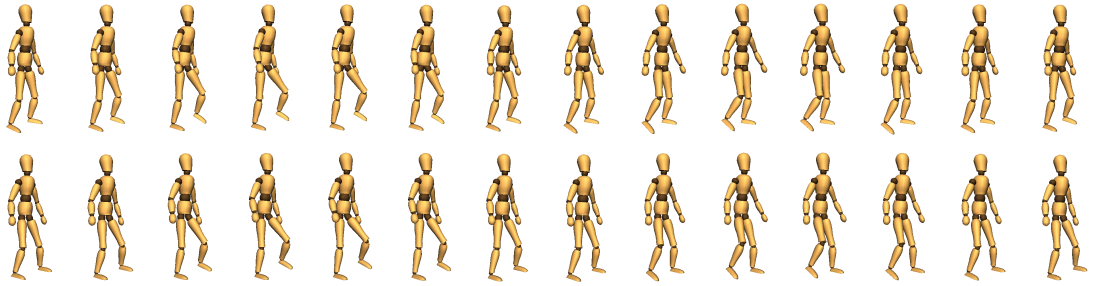
Figure 7.9: Knee-bent sideways walking with hands circling in opposite phase for the "cloud hands" Tai Chi movement. Every third frame.

Motion Capture Imitation

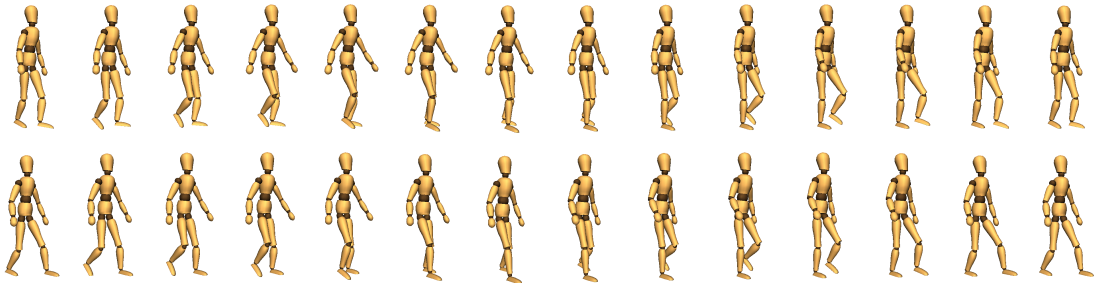
We have developed seven controllers from motion capture data, including four different types of in-place walking (normal, wide-stance, bent-trunk, high-knee), forwards walking, backwards walking, and sideways walking. Figure 7.10 shows comparisons of the original captured motions and the motion resulting from the controllers.



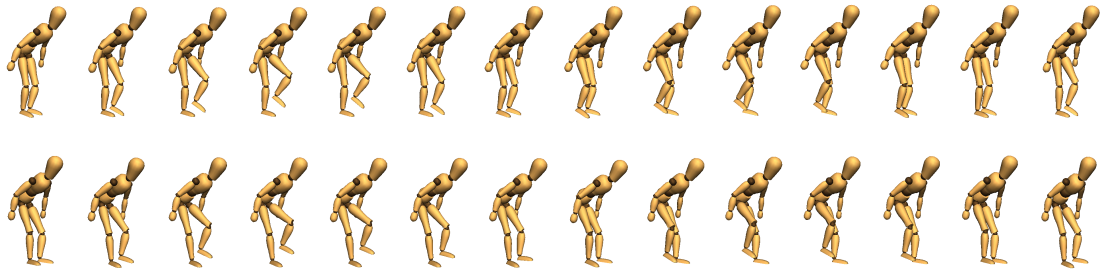
(a) High-knee walking every fourth frame: Upper row - mocap; Lower row - simulation



(b) Wide-stance walking every third frame: Upper row - mocap; Lower row - simulation



(c) Backward walking every third frame, read from right to left: Upper row - mocap; Lower row - simulation



(d) Torso-bent walking every third frame: Upper row - mocap; Lower row - simulation

Figure 7.10: Imitation of motion capture data. Each box compares the original motion (top) and the controller motion (bottom). The boxes from top to bottom: high knee walk; wide stance walk; backwards walk, right-to-left; bent-forwards walk.

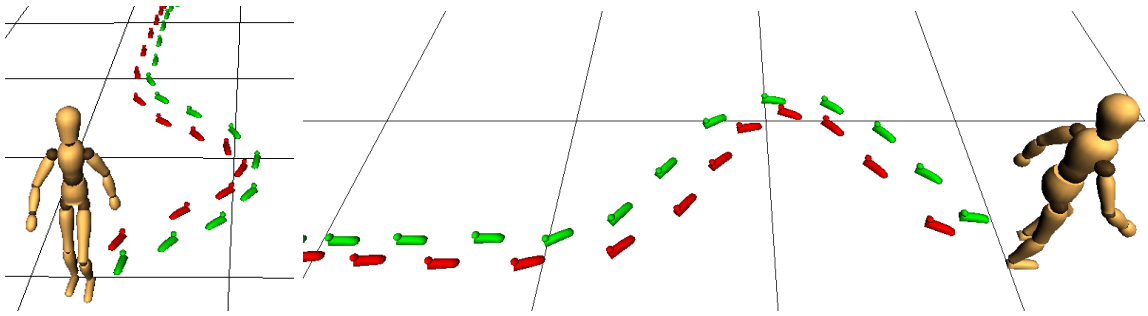
If necessary, we use a small amount of manual tuning to make the tracking-based controller functional. Some of the original in-place stepping gaits exhibit only a small amount of swing-foot clearance during the step. This can cause a failure in the tracking-based controller to lift the swing foot off the ground. A simple correction is to add a constant swing-hip offset so that the swing foot lifts off the ground as desired. Manual tuning of the balance feedback gains for the swing hip, both sagittal and lateral, is sometimes necessary. Section 7.7.3 describes the types of artifacts that are seen when the gains are misadjusted. The in-place high-stepping walk requires use of sagittal balance feedback gains for the stance ankle because of the time spent balanced on the stance foot and the large shift in the CoM caused by the high lift of the swing leg. An unnatural aspect of some of our walking results is that they may fail to properly mimic aspects of the ankle motion and foot toe-off. We speculate that this may resolved with additional tuning of the ankle PD-gains and ankle balance feedback gains. We have not yet tried to replicate running motions from motion capture data, although we are optimistic that this would work.

We speculate that there will be several categories of motions where our motion-capture-to-controller methodology will fail. Acrobatic motions have significant flight phases and therefore rely on accurately achieving specific linear and angular momentum upon takeoff. Our balance control feedback does not provide these. Dynamic motions that do not involve periodic stepping motions are likely to be problematic. Lastly, dynamic motions that do not involve any stepping require a ZMP approach or an approach that can exploit other parts of the body to help maintain balance.

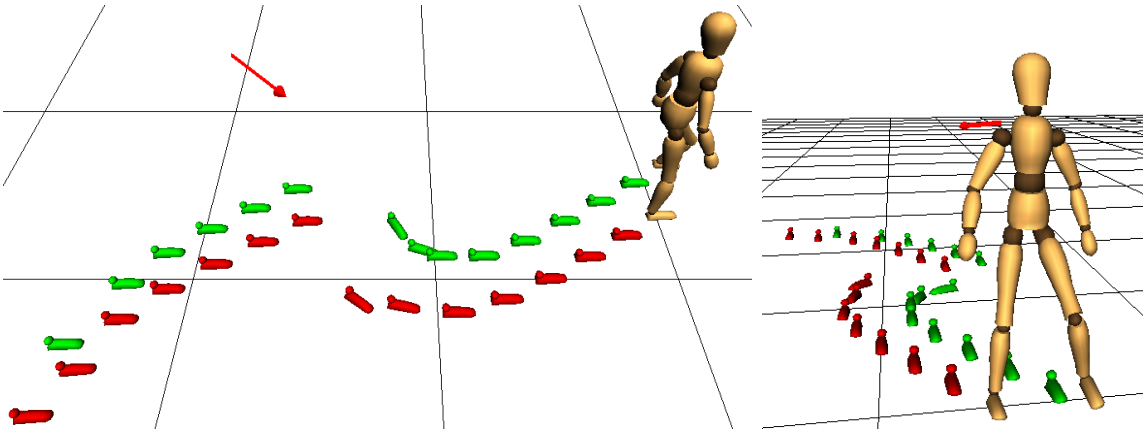
Parameterization

The robust nature of our controllers means that once a controller has been constructed, either manually or from motion capture data, additional control strategies can then be layered on top. High-level controllers can be developed to control walking styles or walking directions. Much of this kind of control can be added in an intuitive fashion as displacements to the target poses or target motions. For example, to add a lateral component to straight line walking, we add an antisymmetric displacement angle to the lateral target hip angles for all the poses in the PCG. If instead we add symmetric displacement angles to each lateral hip angle, this produces a straight walk with altered stance width. The result is shown in Figure 7.11(b), with the addition of an external push. A wide-stance in-place walk reconstructed from motion capture data is made to walk diagonally forward and to the left through the simple addition of offsets to the swing-hip target angles. It then reacts to a large push diagonally to the right.

Controllers can cope with unanticipated gentle slopes and small steps. Steeper slopes or larger steps require adding a displacement $\Delta\theta_{hip} = k\theta_s$ to the target hip angles to insure foot clearance, where θ_s is the angle of the slope. For sufficiently steep



(a) Turning behavior applied to a slow forward walk controller reconstructed from motion capture data. Turning behavior is generated by feeding a sinusoid $0.9 * \sin(\pi t/12)$ in radians as the desired facing direction. Front-view and side-view.



(b) A wide-stance in-place walking reconstructed from mocap is perturbed 0.08 radian sagittally(sagittal $swh+ = 0.08$) and 0.1 radian laterally(left lateral $swh+ = 0.1$) to walk left-forward. Then a right-forward push of magnitude $(-250N, 250N)$ lasting 0.2s is exerted on the chest at phase $\phi = 0.6$. Side-view and front-view.

Figure 7.11: Variations in locomotion, illustrated using footprints.

slopes, the ankle angles also need to be adapted. The result is shown in Figure 7.12.

Turning is generated by modulating the desired facing angle. The desired facing direction in Figure 7.11(a) is varied according to $0.9 * \sin(\pi t/12)$ in radians. The stance hip is then used to achieve the desired facing direction. A light backpack can be worn with an unchanged, vertical torso target pitch. Heavier backpacks need to be accomodated by pitching the torso forward accordingly. Interpolation between gaits can be achieved by interpolating between parameters of the corresponding controller states. For example, a Japanese bow can be generated by interpolating a straight-up walking and a torso-bent walking. Since the balance controller only uses the lower limbs, the upper body is left free for different styles or additional tasks. For example, we can choose to keep the arms straight down or to swing them naturally. Figure 7.9 illustrates Tai Chi cloud hands.

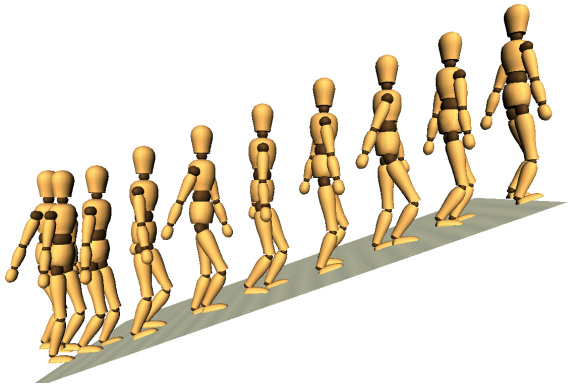


Figure 7.12: Climbing a slope of 20 degrees. 1s lapse.

Robustness

A robust balance controller also means that the locomotion can deal with unexpected environmental disturbances automatically. For the 3D walk controller given in Table 7.1, the largest recoverable pushes as measured in eight evenly-sampled directions are $(0, 340)$, $(230, 230)$, $(330, 0)$, $(220, -220)$, $(0, -270)$, $(-190, -190)$, $(-240, 0)$, $(-190, 190)$, as illustrated in Figure 7.13, where each pair defines the (lateral,sagittal) push magnitudes in Newtons. The pushes are applied at chest height at a phase angle of $\phi = 0.1$ and have a duration of 0.4s. These numbers are comparable with specially developed push recovery controllers described in [Kudoh et al. 2006], which are only demonstrated in the sagittal plane and are computed offline using quadratic programming. We also tested the robustness with respect to kinematic and dynamic model variations. For example, we have increased the femur length by 10% for the walking gait defined in Table 7.1 while maintaining both the style and stability of the gait. Larger changes can be accomodated, first resulting in a change of style while still being robust. Dynamic parameters such as the mass and inertia have minimal effects as long as the PD controller gains are scaled accordingly. Some gaits are particularly sensitive to some parameters. For example, the fast running gaits tend to be sensitive to the balance feedback gains. Stairs can cause problems because the controllers cannot "see" an upcoming step, and the resulting toe stub or ill-placed foot can cause a fall.

Integration with Interactive Balancing

An integration of the dynamic in-place and stepping balancing system described in Chapter 6 with the SIMBICON framework is demonstrated in http://www.cs.ubc.ca/∼kkyin/animation/MultiStepping.mov. SIMBICON enables the multistepping strategy to be realtime.

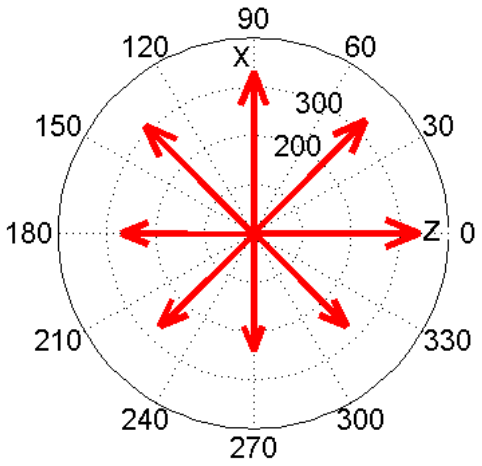


Figure 7.13: The largest pushes from eight representative directions at the chest level recoverable when applied at $\phi = 0.1$ for 0.4s. Character facing to the right(positive z axis).

Feedback Error Learning

Feedback error learning has been successfully applied to the upper-body joints and the virtual torso torque for all manually-designed FSM controllers. We limit the maximum feedforward torques to be $|\Delta\theta k_p|$, where k_p is the PD spring constant and $\Delta\theta = 0.2$ radians. The feedforward torque will thus be capable of eliminating oscillations of approximately 0.2 radians in magnitude. We experiment with various resolutions for representing the feedforward torque, using phase bins that correspond to $\Delta t_\phi \in [0.5, 25]\ ms$. For the 3D walk controller given in Table 7.1, we test several different resolutions and computed the energy ratio of the feedback torques to the feed-forward torques, $r = \frac{\int ||fb||}{\int ||ff||}$ after FEL. For $\Delta t_\phi = 0.5ms, r \approx 2.5\%$; $\Delta t_\phi = 5ms, r \approx 3.5\%$; $\Delta t_\phi = 25ms, r \approx 14\%$. Because θ_{d0} is discontinuous for the lower-body joints for the manually-designed controllers, we do not apply FEL to these joints. The large discontinuities in the desired joint angles that occurs upon transitioning to a new FSM state are not suitable for modeling directly using a feed-forward torque. This can be circumvented by treating the output of a simulation as being the equivalent of motion capture data, and applying the strategy that we shall describe next.

Feedback error learning can also be applied to controllers that track motion capture data. FEL can be applied directly to the upper-body joints and the virtual torso torque. Applying FEL to the lower limbs requires two adaptations to the basic FEL learning process. First, for all joints using balance feedback, the joint angle tracking torques need to be decoupled from the balance-feedback torques. Therefore it is essential to apply FEL based only on the component of the torque that is used to track

the joint angle target trajectory, and *not* the component that is added to achieve balance control. Second, despite the use of smooth target-angle trajectories, there remains a discontinuity at the instant of stance-leg/swing-leg exchange. To accomodate this, we adapt the desired trajectory towards the realized simulation trajectory using a displacement trajectory. The displacement trajectory $\delta\theta$ is initialized to zero and is modified over time according to $\delta\theta' = (1-\alpha)\delta\theta + \alpha(\theta - \theta_d)$. Each of $\delta\theta, \theta, \theta_d$ are functions of phase and are modeled using phase bins in the same way as feed-forward torques. It can be seen that $\delta\theta$ remains unchanged when it correctly predicts the tracking error $\theta - \theta_d$.

In the video http://www.cs.ubc.ca/~kkyin/animation/Yin_SIG07.mov, we show several comparisons between walks based on motion capture data, simulated using (a) feedback control alone, and (b) using combined feedback and feedforward control. The latter motions exhibit smoother, more stable motion while having PD constants k_p, k_d that are the same or lower.

7.7.3 Setting the Balance Feedback Gain Parameters

c_d, c_v are usually within the range of $[0, 1]$. For our basic 3D walk controller we use $c_d = 0.5$ and $c_v = 0.2$ for the swing hip in all states, in both the coronal and sagittal planes. When changing parameter values, implementors may observe phenomena such as period doubling [Vakakis and Burdick 1990; Buhler and Koditschek 1988]. We use a stability analysis to provide an indication of the sensitivity of the results with respect to some of the parameters. Beginning from a fixed initial state, we change one selected parameter across all four states to find its viable range, while fixing all other parameters to their nominal values. The stable range of parameters is: $[-0.71, 1.4]$ for c_d and [0.03,0.59] for c_v in the sagittal plane; $[-1.29, 1.13]$ for c_d and [-0.06,0.48] for c_v in the coronal plane. For most operating points within these ranges, stable limit cycles can be achieved. For operating points near the upper limits, period doubling and chaotic behavior develop on occasion. When c_v is below the lower limit, the velocity of the character accumulates until it falls. When c_v is above the upper limit, usually the character will rock back and forth(or left and right) with increasing amplitudes until the oscillations destroy the walking.

7.7.4 Limitations

Figure 7.14 is the ground reaction force recorded from our simulation of a normal walk reconstructed from a motion capture example. As we can see there are unrealistically large forces when the stance leg and swing leg switch roles. However, other parts of the GRF pattern are qualitatively similar to data reported in [Medved 2001; Rose and Gamble 2006], which give the recorded GRF from human walking. To reduce the impact forces for additional fidelity in imitating human walk cycles, several

issues need to be addressed. The current controllers do not explicitly model some phases of motion such as double-stance and toe-off. Extra states dedicated to these particular phases are desirable. Velocity matching is needed to reduce impacts in human locomotion [Hodgins et al. 1995]. A better foot ground contact model that has compliance to imitate human soles and shoes would likely help. Lastly, a foot model having internal joints is likely needed if a fully natural toe-off behavior is desired.

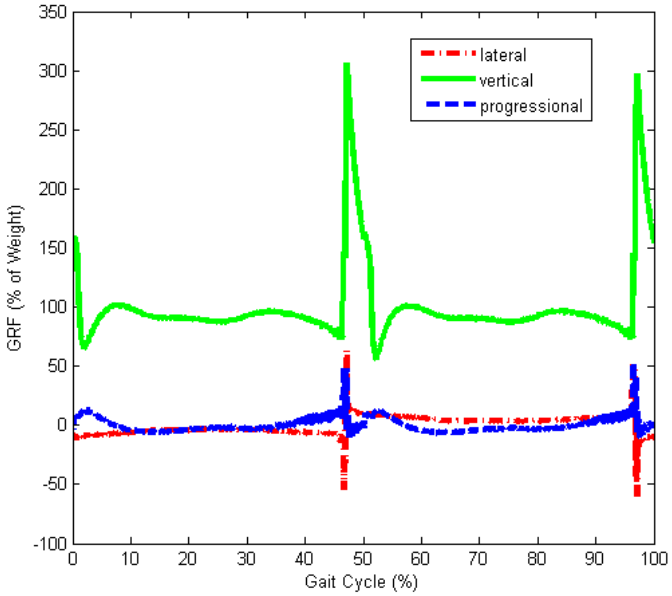


Figure 7.14: Ground reaction force recorded from simulation of a normal walk reconstructed from a motion capture example.

The pipeline for producing controllers from motion capture data is not fully automated in that we still manually tune the required feedback gain constants, and has only been tested on styles of walking. The current gaits are not optimized for energy efficiency. We do not model the reaction-time delays of human motion. As a result, some of our motions are stable in a way that human motions may not be, such as being able to accomodate unanticipated $20cm$ downwards steps.

The available suite of mathematical tools for the stability analysis of high-dimensional, non-linear dynamical systems is limited [Strogatz 1994]. Two practical options for analysis are to work with a simplified version of the system dynamics, or to rely on simulation-based experiments. We have chosen the latter option.

7.8 Discussion

The control of bipedal locomotion is a challenging problem. The need in animation to model multiple gaits, stylized motions, reaction to variable terrain, and reactions

to external forces exacerbate this challenge. The framework presented in this paper addresses many of these challenges. We have further shown how to develop a variety of walking controllers from motion capture data and how to implement feedback error learning to achieve motions that are driven by feed-forward torques and low-gain feedback.

There are a large number of directions that can be pursued. We wish to develop libraries of 'downloadable skills' that can be shared. This requires file formats for exchanging controllers which describe both the controller itself and its basin of attraction [Faloutsos et al. 2001]. We wish to apply the control schemes to humanoid robots. Basic locomotion skills should be integrated with other skills that let the simulated characters interact with their environment in a rich variety of ways. Methods are needed for planning motions using the controllers we have developed.

Autonomous characters (or robots) should exhibit more sophisticated anticipation and response with respect to its environment. This includes better adaptation to the terrain as well as navigating among moving objects and people. In the current work we have explored a model of simply switching between controllers. A more sophisticated scheme could exploit the continuous parameterizations obtained by interpolating between controllers. Alternatively, a system could be developed to identify and learn (through optimization) new acyclic transition motions that would lead to more agile behaviors. We wish to more thoroughly examine the effect of differences between the physical parameters of the motion capture subject and those of the simulation model.

state	Δt	c_d	c_v	*tor*	*swh*	*swk*	*swa*	*stk*	*sta*
	walk								
0,2	0.30	0.00	0.20	0.0	0.40	-1.10	0.20	-0.05	0.20
1,3	fc	2.20	0.00	0.0	-0.70	-0.05	0.20	-0.10	0.20
	in-place walk								
0,2	0.30	0.00	0.40	0.0	0.62	-1.10	0.20	-0.05	0.20
1,3	fc	1.55	0.00	0.0	-0.10	-0.05	0.20	-0.10	0.20
	fast walk								
0,2	0.27	0.00	0.20	-0.1	0.73	-1.83	0.20	-0.05	0.20
1,3	fc	2.00	0.00	-0.1	-0.70	-0.05	0.20	-0.10	-0.06
	highstep walk								
0,2	0.30	0.00	0.20	0.0	1.00	-2.40	0.20	-0.05	0.20
1,3	fc	2.00	0.00	0.0	-0.70	-0.05	0.20	-0.10	0.20
	half bent walk								
0,2	0.23	0.00	0.20	-0.2	0.62	-1.10	0.00	-0.05	0.00
1,3	fc	0.60	0.00	-0.2	-0.10	-0.05	0.00	-0.10	0.00
	bent walk								
0,2	0.30	0.00	0.20	-0.6	0.80	-1.10	0.00	-0.05	0.00
1,3	fc	0.60	0.00	-0.6	-0.10	-0.05	0.00	-0.10	0.00
	crouch walk								
0,2	0.30	0.00	0.20	-0.2	1.10	-2.17	0.62	-0.97	0.44
1,3	fc	2.20	0.00	-0.3	-0.70	-0.05	0.20	-0.92	0.44
	scissor hop								
0,2	0.27	0.00	0.77	-0.2	0.70	-0.58	0.20	-0.05	0.09
1,3	fc	0.11	0.01	-1.0	-0.82	-0.27	0.20	-0.10	0.12
	backwards leaning backwards walk								
0,2	0.22	0.00	0.28	0.2	0.37	-1.41	0.00	-0.05	0.00
1,3	fc	0.60	0.00	0.3	-0.10	-0.05	0.00	-0.10	0.00
	fast run								
0,1	0.15	0.00	0.20	-0.2	1.08	-2.18	0.20	-0.05	0.27
	run								
0,2	0.21	0.00	0.20	0.0	0.80	-1.84	0.20	-0.05	0.27
1,3	0.00	0.00	0.20	-0.2	1.08	-2.18	0.20	-0.05	0.27
	skipping gait								
0,4	0.19	0.00	0.40	0.0	1.04	-1.75	0.20	-0.19	0.20
1,5	0.12	0.00	0.40	0.0	2.25	-2.18	0.20	-0.05	-1.60
2,6	0.26	0.00	0.04	0.0	2.44	-2.09	0.20	-0.05	0.20
3,7	fc	0.18	0.37	0.0	-0.46	-0.05	0.20	-0.10	0.20
	3D walk								
0,2	0.3	0.5	0.2	0	0.5	-1.1	0.6	-0.05	0
lat		0.5	0.2	0	0	0	0	0	0
1,3	fc	0.5	0.2	0	-0.1	-0.05	0.15	-0.1	0
lat		0.5	0.2	0	0	0	0	0	0
	3D run								
0,1	0.3	0.5	0.2	0	0.5	-1.1	0.6	-0.05	0
lat		0.5	0.2	0	0	0	0	0	0

Table 7.1: 2D and 3D locomotion parameters for the periodic, left-right symmetric gaits. The columns from left to right represent the state numbers, state dwell duration, position and velocity balance feedback coefficients, and the torso, swing-hip, swing-knee, swing-ankle, stance-knee, and stance-ankle target angles. All angles are expressed in radians. The 2D and 3D runs have only two states.

Chapter 8

Conclusions

This thesis has outlined several ways to generalize character animation from motion capture data. Depending on the application, some of them are more successful than others. Kinematic modelling is more successful where the necessary knowledge and insights are not available to help reconstruct a controller for dynamic simulation. The main issues are how to parametrize or index the precaptured motions, properly select matching examples, and seamlessly adapt and blend those example motions. Dynamic modelling is more successful in permitting wider generalization and bigger user interactions that guarantees physical realism, and requires less example data.

8.1 Contributions

FootSee is a novel interface for interactive avatar control and low-cost performance-driven animation, using a foot pressure sensor pad and pre-captured pressure and motion databases. It is intuitive, non-intrusive and reasonably robust. We believe FootSee is a promising technique that provides an intuitive and easy-to-use interactive interface for many types of applications, including interactive video games, avatar control, sports training, and performance-driven animation.

Our data-driven kinematic balancing system is fast and effective, even when using a relatively small motion database. It is useful for interactive video games, fast balance behavior choreography, autonomous avatar control in virtual reality applications, and reference trajectory formation for humanoid robots. It successfully generated motions that respond to multiple pushes, as well as to single pushes, although the motion database only contains balance behaviors under single pushes. Multiple strategies, including arm rotation strategy which has not been successfully reproduced in our dynamic approaches, are demonstrated. The results show that a data-driven approach can be quite powerful, when accurate physical realism is not required, or when the motor task is beyond our understanding to implement dynamically.

Motor control is one of the major challenges in physically based human simulation. To our knowledge, our work is the first that tries to address this problem by explicitly

incorporating human neuromotor control models into the human simulation system. We test our approach with motion perturbation tasks on motion capture data, and the results are promising. Applications such as interactive video games can thus benefit from an enriched motion repertoire.

Our dynamic in-place and stepping balancing system is omnidirectional, while related work has most often focussed on modeling control in response to forward pushes in the sagittal plane. The control models we develop and demonstrate can respond in realtime to a wide range of push magnitudes and directions, with multiple strategies.

In the dynamic locomotion framework SIMBICON, we integrate and build on previous insights to develop a simple new strategy for the control of balance during locomotion. We show that this can be used to develop controllers for a wide variety of 2D and 3D biped gaits. To our knowledge, we are the first to demonstrate a large set of integrated, physically-simulated bipedal skills, including many styles of walking, omni-directional push-recovery while walking, running, stylized running (scissor hop), and skipping. In addition, we develop and demonstrate controller-based imitation of motion-captured gaits which exhibit robust balancing behavior. Lastly, we demonstrate that feedback error learning can be used to produce anticipatory, low-gain locomotion control. The simple framework opens the door to developing significantly wider sets of locomotion skills for physically-simulated characters and bipedal robots.

8.2 Future Work

There are many promising directions for future research.

For the FootSee system described in Chapter 3 , we would like to combine other non-intrusive sensing techniques with FootSee. For example, a camera can probably see the arm motions better than FootSee, while FootSee can resolve many ambiguities that a camera cannot. Chai and Hodgins [2005] demonstrated an interesting performance driven system with two cameras and six markers. Our initial intention for the foot pressure sensor pad was to extract the dynamic force-ground interaction information to aid the dynamic modelling process of balance and walking controller, or to do dynamic performance-driven animation (cell *A1* in Table 1.1). This remains an interesting direction to pursue.

For the data-driven balancing system in Chapter 4, we plan to explore more interactive motor task synthesis using this kind of data-driven approach, with dynamic indices and constraint-respecting transformations.

Our approach to using insights into human motor control in Chapter 5 is very simple: only muscle-tendon feedback is considered; spinal and supraspinal feedback pathways are missing. Longer latency feedback pathways, which are also more complex and less well understood, can realize reflex-like rapid movements and phase-

dependent functionally correct responses. Although the SIMBICON framework in Chapter 7 exhibits some degree of reflex-like stepping, the richness cannot compare with the response repertoire of real humans, and the responses are not always appropriate for different stages of a gait cycle.

Synergy learning (Section 2.3.2) is potentially a very useful concept that not only can accelerate the learning process, but also makes the synthesized motion less robotic and more human like. However, how to learn, represent and exploit synergy is not clear. A starting point may be to analyze torques computed by inverse dynamics on motion captured data.

The SIMBICON framework given in Chapter 7 opens significant opportunities for future study. We wish to develop libraries of controllers that can be shared. One way to do this is to enable formats for exchanging controllers which describe not only the controller itself but also its basin of attraction. This moves towards a workable implementation of downloadable skills for autonomous characters. We also wish to apply the control schemes to humanoid robots.

Autonomous characters (or robots) should exhibit more sophisticated anticipation and response with respect to its environment. This includes better adaptation to the terrain as well as navigating among moving objects and people. In SIMBICON we have explored a model of simply switching between controllers. A more sophisticated scheme could exploit the continuous parameterizations obtained by interpolating between controllers. Alternatively, a system could be developed to identify and learn (through optimization) new acyclic transition motions that would lead to more agile behaviors. We wish to more thoroughly examine the effect of differences between the physical parameters of the motion capture subject and those of the simulation model.

In this thesis we basically use a reactive change-of-support strategy to reposition the swing leg for balance control. Momentum control, as briefly mentioned in Section 2.2.1, and in recent work of Lee and Goswami [2007], can capture the critical angular behaviours of human movements, such as windmilling effects of the trunk and arms when we are about to lose balance. SIMBICON can serve as a base layer of control, on top of which the momentum control can sit.

In both dynamic balancing systems, the swing leg can collide with the stance leg during the course of balance recovery. This behavior can likely be predicted and corrected for.

Quantifying the quality of motions, either by user studies or computational techniques[Reitsma and Pollard 2003; Harrison et al. 2004; Ren et al. 2005], will help to justify our results and compare with results from other research groups.

Appendix A

More About Balance Indices

A.1 GRF and CoP

Here we prove that the tangential (to the ground) torque generated by the normal ground reaction forces is zero. In Figure 2.4(a) $\mathbf{f}_{ni} = (0, f_{ni_y}, 0)^T$ denotes the normal force at ground point $\mathbf{q}_i = (q_{i_x}, 0, q_{i_z})^T$, then the definition of the CoP $\mathbf{p} = (p_x, 0, p_z)^T$ is as follows:

$$\begin{aligned} p_x &= \frac{\sum q_{i_x} f_{ni_y}}{\sum f_{ni_y}} \\ p_z &= \frac{\sum q_{i_z} f_{ni_y}}{\sum f_{ni_y}} \end{aligned} \tag{A.1}$$

So the resultant tangential torque of these normal forces at $\mathbf{p}$ is:

$$\mathbf{t}_t = \sum (\mathbf{p} - \mathbf{q}_i) \times \mathbf{f}_{ni} = \begin{bmatrix} \sum -(p_z - q_{i_z}) f_{ni_y} \\ 0 \\ \sum (p_x - q_{i_x}) f_{ni_y} \end{bmatrix} = (0, 0, 0)^T \tag{A.2}$$

A.2 ZMP

The ZMP is defined as that point on the ground at which the net moment of the inertial forces and gravity forces has no component along the horizontal axes. The fundamental reason for caring about the horizontal net moment can be explained by D'Alembert's Principle. Intuitively, we know from the last section that ground reaction force has no tangential moment at the CoP, and mathematically the ZMP and the CoP are equivalent ([Goswami 1999]), so inertial forces and gravity forces cannot have horizontal moment either at the CoP, if the system is in dynamic equilibrium.

D'Alembert's Principle states that the sum of all forces: external forces and inertial forces, inside a dynamic system is equal to zero. Or alternatively the virtual work done by external forces to a system is equal to the virtual work done by inertial forces. We can write out the rotational dynamic equilibrium about any stationary reference

point **o** as follows, ***when the gravity and the GRF are the only external forces***:

$$\mathbf{t} + \mathbf{op} \times \mathbf{f} + \sum \mathbf{oc}_i \times m_i \mathbf{g} = \sum \dot{\mathsf{H}}_i + \sum \mathbf{oc}_i \times m_i \mathbf{a}_i \tag{A.3}$$

where m_i is the mass of link i; $\mathbf{c}_i$ is the Center of Mass (CoM) location; $\mathbf{a}_i$ is the CoM linear acceleration. $\dot{\mathsf{H}}_i = \mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i$ is the rate of change of angular momentum about the CoM of the i^{th} segment, where α_i is the angular acceleration and ω_i is the angular velocity. $\mathbf{p}$ is the CoP. $\mathbf{t}$ is the resultant ground reaction torque at $\mathbf{p}$, and $\mathbf{f}$ if the resultant ground reaction force at $\mathbf{p}$.

If we take $\mathbf{p}$ as $\mathbf{o}$, Equation A.3 can be reduced to

$$\sum \dot{\mathsf{H}}_i + \sum \mathbf{pc}_i \times m_i \mathbf{a}_i - \sum \mathbf{pc}_i \times m_i \mathbf{g} = (0, *, 0)^T \tag{A.4}$$

We see the CoP actually satisfies the ZMP definition, thus they are equivalent. In robotics, people usually use the ZMP, while in biomechanics (Section 2.3.1) people usually use the CoP. From now on we will denote the ZMP as $\mathbf{z}$, and the CoM of each link as $\mathbf{p}_i$. We rewrite the ZMP equation for systems with no external forces other than gravity and GRF acted upon as:

$$\sum_i (m_i(\mathbf{p}_i - \mathbf{z}) \times \mathbf{a}_i + \mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i) - \sum_i m_i(\mathbf{p}_i - \mathbf{z}) \times \mathbf{g} = (0, *, 0)^T \tag{A.5}$$

Solving $\mathbf{z}$ with $\mathbf{g} = (0, -9.8, 0)^T$ gives

$$\begin{aligned} z_x &= \frac{\sum_i m_i(p_{ix}(a_{iy} + 9.8) - p_{iy}a_{ix}) + \sum_i (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i)_z}{\sum_i m_i(a_{iy} + 9.8)} \\ z_z &= \frac{\sum_i m_i(p_{iz}(a_{iy} + 9.8) - p_{iy}a_{iz}) - \sum_i (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i)_x}{\sum_i m_i(a_{iy} + 9.8)} \end{aligned} \tag{A.6}$$

When there is a perturbation force $\mathbf{f}$ acting at $\mathbf{s}$, the above ZMP equations change to:

$$\begin{aligned} z_x &= \frac{\sum_i m_i(p_{ix}(a_{iy} + 9.8) - p_{iy}a_{ix}) + \sum_i (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i)_z + f_x s_y - f_y s_x}{\sum_i m_i(a_{iy} + 9.8) - f_y} \\ z_z &= \frac{\sum_i m_i(p_{iz}(a_{iy} + 9.8) - p_{iy}a_{iz}) - \sum_i (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i)_x + f_z s_y - f_y s_z}{\sum_i m_i(a_{iy} + 9.8) - f_y} \end{aligned} \tag{A.7}$$

We can also write the above equations in terms of COM position, velocity, acceleration, and the moment around COM.

$$\begin{aligned} z_x &= \frac{mp_x(a_y + 9.8) - mp_y a_x + t_z + f_x s_y - f_y s_x}{m(a_y + 9.8) - f_y} \\ z_z &= \frac{mp_z(a_y + 9.8) - mp_y a_z - t_x + f_z s_y - f_y s_z}{m(a_y + 9.8) - f_y} \\ \mathbf{t} &= \sum_i (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i) + \sum \mathbf{pp}_i \times m_i \mathbf{a}_i \end{aligned} \tag{A.8}$$

The denominator of the above equations is actually the Y component of the GRF, which should be greater than zero. ZMP is the point where the horizontal moment of external forces and inertial forces balance. The ZMP concept is very important to robot balance control. We want to emphasize that ZMP inside foot boundary is a necessary but not sufficient condition for balance. For a physically plausible motion, ZMP is always inside or on the edge of the foot boundary, because according to D'Alembert's principle, a dynamic system is always in dynamic equilibrium.

A.3 FRI

The FRI point is a point on the ground at which the resultant moment of the force/torque impressed on the foot is normal to the ground surface [Goswami 1999]. Impressed force/torque means the force and torque at the ankle joint, other external forces on the foot, plus the weight of the foot, but not the GRF. Suppose there is only one foot in contact with the ground, and the body index for the foot is 0. We denote the ankle torque as τ_0, ankle force as $\mathbf{f}_0$, ankle joint location as $\mathbf{a}$, foot CoM as $\mathbf{o}_0$, foot mass as m_0. Then FRI point $\mathbf{r}$ is defined mathematically as the point on ground that satisfies:

$$\tau_0 + \mathbf{ra} \times \mathbf{f}_0 - \mathbf{ro}_0 \times m_0 g = (0, *, 0)^T \tag{A.9}$$

Using D'Alembert's principle we can calculate FRI as follows:

$$\begin{aligned} r_x &= \frac{9.8 m_0 p_{0_z} + \sum_{i=1}^{n} m_i (p_{i_x}(a_{i_y} + 9.8) - p_{i_y} a_{i_x}) + \sum_{i=1}^{n} (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i)_z}{9.8 m_0 + \sum_{i=1}^{n} m_i (a_{i_y} + 9.8)} \\ r_z &= \frac{9.8 m_0 p_{0_x} + \sum_{i=1}^{n} m_i (p_{i_z}(a_{i_y} + 9.8) - p_{i_y} a_{i_z}) - \sum_{i=1}^{n} (\mathsf{I}_i \alpha_i + \omega_i \times \mathsf{I}_i \omega_i)_x}{9.8 m_0 + \sum_{i=1}^{n} m_i (a_{i_y} + 9.8)} \end{aligned} \tag{A.10}$$

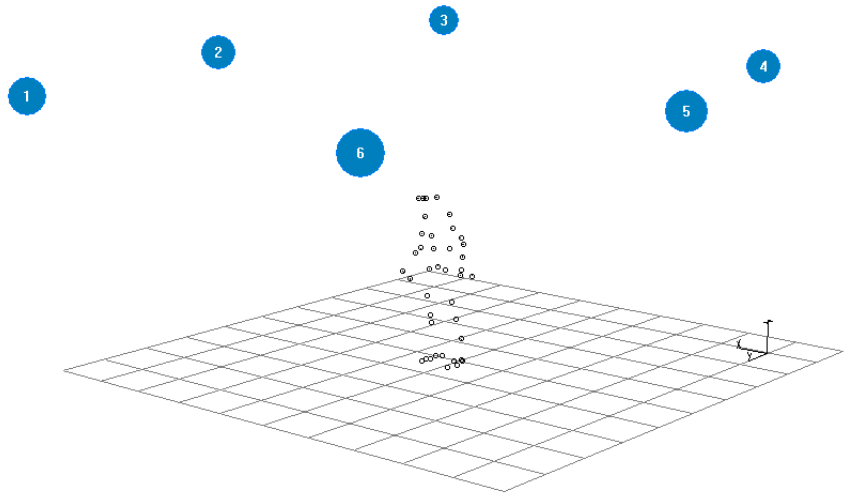
FRI and ZMP are closely related. From Equation A.6 we can easily get Equation A.10 by setting $a_0 = \alpha_0 = \omega_0 = 0$. Unlike the ZMP, the FRI can be outside of the support polygon. It is where the net ground reaction force would have to act to keep the foot stationary. Intuitively, we can think of the FRI as the virtual ZMP when the foot is infinitely large and static. It has the following properties:

- The FRI point indicates the occurrence of foot rotation.
- The location of the FRI point indicates the magnitude of the unbalanced moment on the foot.
- The FRI point indicates the direction of foot rotation.
- The FRI point indicates the stability margin of the robot.

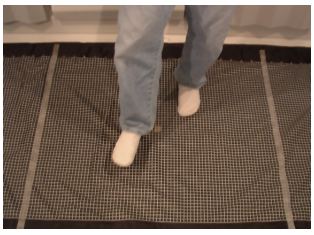
Appendix B

Hardware used in Our Experiments

B.1 Equipment



(a)



(b)

Figure B.1: (a) Typical setup of the Vicon6 motion capture system for full body motion capture. (b) XSensor pressure sensor bed pad.

For motion capture, we use the Vicon6 motion capture system[1]. It consists of a Vicon6 datastation and six MCam2 infrared cameras. Since the capture resolution reduces when the capture frequency increases, we use 60 or 120 Hz for full body motion capture to get the largest resolution (3160×1024 for 60Hz). A typical setup for full body motion capture is shown in Figure B.1(a).

For foot-ground pressure capture, we use the XSensor[2] pressure sensor bed pad (Figure B.1(b)).The bed pad is capable of capturing an area of 0.8m×2m at 6 Hz, or 0.8m×1m at 12 Hz, with a spatial resolution (center-to-center spacing of individual pressure sensors) of 0.5 inch. There are four pressure ranges where we can get calibrated data: 0-30 psi, 0-50 psi, 0-80 psi, 0-100 psi. According to our experience, 0-50 psi is good for normal 2 feet balancing, walking etc. 0-80 psi is good for balancing on one foot, jumping etc. 0-100 psi is good for high dynamic motions. 0-30 psi is good for very static motions and can give better accuracy. The foot pads are capable of capturing two 7.5W North American shoe-size (23.5 cm) feet, at 50 Hz onto a smart media card, or 20 Hz onto a hard drive directly. The spatial resolution is 7 mm. It can return calibrated data for the 0-80 psi pressure range. All the pads can return raw pressure data in 0-255 byte format as well.

B.2 Synchronization between Vicon and XSensor

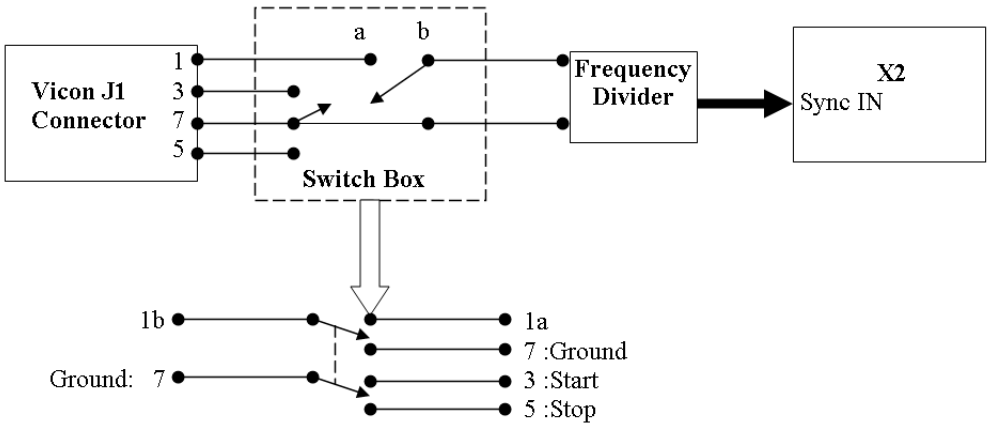


Figure B.2: Sync circuit

The original function of the Vicon J1 connector is to allow the remote control of data capture from external switches or photoelectric sensors ([Vicon]). Connecting Start (pin 3) or Stop (pin 5) to Ground (pin 7) will initiate the selected function.Pin 1

[1]Vicon Motion Systems Ltd: http://www.vicon.com/

[2]XSensor Technology Corporation: http://www.xsensor.com/

generates a negative going TTL gated reference signal, which is aligned to the camera Horizontal Synchronization (HD) signal and present when cameras are performing. That is to say, we have a 60 Hz signal coming out of Pin 1 if we capture at 60 Hz. The first generation of our synchronization circuit directly feeds this signal to a homemade frequency divider, and a 12 Hz sync signal is carried into the X2 Sync IN socket by a 3.5mm stereo audio jack ([XSensor]).

The problem with the first generation circuit is that although motion and pressure frames are synchronized, the start time of the two captures are not. This is because the sync signal from Pin 1 is active as soon as we start the "live monitor" in Vicon Workstation. We did the start time alignment by manual synchronization. At the beginning of each trial, we asked the subject to do one or two stomps on the pressure pad. Later we watch the motion and the pressure data to synchronize the stomps.

Figure B.2 shows our second generation sync circuit. What we want is to activate the sync signal only when we start the actual Vicon capture. We combine the standard Vicon remote/external triggering switch box with our first generation sync circuit, using an on-off double pole double throw rocker switch.

For future work, a third generation synchronization will get rid of the frequency divider circuit. A standard serial port will carry the 60 Hz sync signal into the computer and interrupt our software whenever a pulse arrives. Our software will then do a more flexible frequency manipulation and call the XSensor API function *Sample()* when needed.

B.3 XSensor Setup

In setting up and using the XSensor hardware and in testing our own software, we encountered several issues.

- The XSensor electronics box is able to control only one task at a time, with first priority given to the XSensor software, i.e. when the XSensor program is open, regardless of whether or not it is in recording mode, other programs which try to access the X2 electronics box through the XAPI.dll may act unpredictably.

- The XSensor pad is sensitive to electromagnetic conductive surfaces. The area around the pad should be cleared in order to ensure that noise from external devices does not interfere with data capture. The amount of noise that may appear from devices such as USB hubs can be rather significant. The metal surface of the floor in the lab will also impact pressure readings; avoid this by using a barrier between the floor and the bottom of the pad (e.g. a large piece of cardboard, hard carpeting etc.).

Appendix C

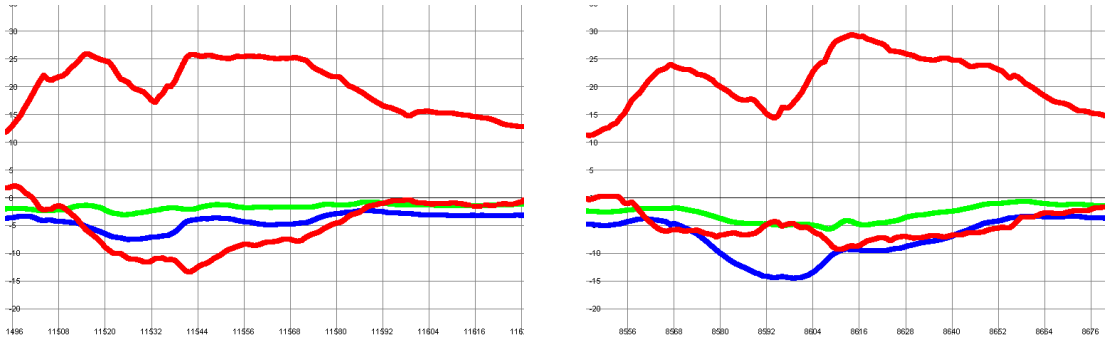
Inverse Kinematics Algorithm for FootSee

As stated in Chapter 3, we developed an analytical inverse kinematics (IK) algorithm to modify the stepping motions selected from the database to match the user's step length when a large position error is detected. The IK has the following steps:

1. The root orientation and planar (x, z) position remain unchanged.
2. The hip joint angles are modified to meet stepping length changes.
3. After the hip joint angles are edited, some constraints will be violated, causing problems such as rotation of the foot during stance, and ground penetration during swing. The ankle and knee joint angles are modified to keep these constraints satisfied.

The rationale for the design is as follows. The step length, step direction and step height are mainly a function of the orientations of the pelvic girdle, the left and the right hips, and the left and the right knees. In our experiments, the user is always facing the same direction, and so the orientation of the pelvic girdle (the root orientation) does not change much. We thus leave the pelvic orientation fixed in our IK.

Next to the shoulder joint, the hip joint is the most movable of all joints. It is a ball-and-socket type of joint (3 DoF). The knee joint is primarily a hinge type of joint, combined with a small amount of gliding and rolling [Moore and Dalley 1999]. It is usually treated in computer graphics as a 1-DoF joint allowing only flexion and extension. Although our skeleton model treats all joints as 3-DoF joints, the 2 extra joint angles of the knees remain small all the time. Because we consider only flat terrain, the knee flexion-extension characteristics for different steps do not vary greatly in our experiments. Figure C.1 shows the joint angles of two steps we captured. One is a right forward stepping, the other is a right side stepping. The patterns for knee angles are quite similar: a flexion in toe-off, followed by an extension in strike,



(a) right forward stepping (b) right side stepping

Figure C.1: The joint angles of the right leg for two steps. The horizontal coordinate is the frame number. The vertical coordinate is the joint angle. In both figures, the top red curve is the knee joint angle. The bottom red curve is the hip flexion-extension. The blue curve is the hip abduction-adduction. The green curve is the hip axial rotation. Please refer to Figure 3.5(b) for the joint angles' corresponding rotation axes which are coloured accordingly.

followed by a flexion in opposite toe-off, followed by an extension in opposite strike [Rose and Gamble 1994]. So basically there are two peaks in the knee angle curves. The hip angles, however, vary with the stepping length and direction more directly. In side stepping, there is more abduction-adduction; while in forward stepping there is more flexion-extension. Therefor, the hip angles are chosen to adapt the step length, and ankle and knee angles are used to maintain constraints.

Based on the above assumptions, we designed the fast IK algorithm shown in Figure C.2. In Figure C.2(a), we know the initial joint positions of the leg, thus we can compute the hip-ankle vector $\mathbf{a}$, and the projected hip-ankle vector $\mathbf{b}$ ($\mathbf{a}$ projected onto the horizontal plane through the ankle). Suppose we want to shift the ankle position by $\mathbf{c}$ to get to a new location estimated from the new pressure data, keeping the knee joint angle unchanged and allowing the hip position to only move vertically (which is equivalent to the first assumption). The new hip-ankle vector $\mathbf{e}$ can be computed, because $\mathbf{d}$ can be computed from $\mathbf{b}$ and $\mathbf{c}$, and the length of $\mathbf{e}$ equals the length of $\mathbf{a}$. With $\mathbf{a}$ and $\mathbf{e}$ available, we compute the quaternion $\mathbf{q}$ that rotates $\mathbf{a}$ to $\mathbf{e}$.

$$\mathbf{q} = (\mathbf{k}\sin(\theta/2), \cos(\theta/2)) \tag{C.1}$$

where

$$\theta = \arccos(\frac{\mathbf{a}\cdot\mathbf{e}}{\|\mathbf{a}\|\|\mathbf{e}\|}) \tag{C.2}$$

and

$$\mathbf{k} = \frac{\mathbf{a}\times\mathbf{e}}{\|\mathbf{a}\times\mathbf{e}\|} \tag{C.3}$$

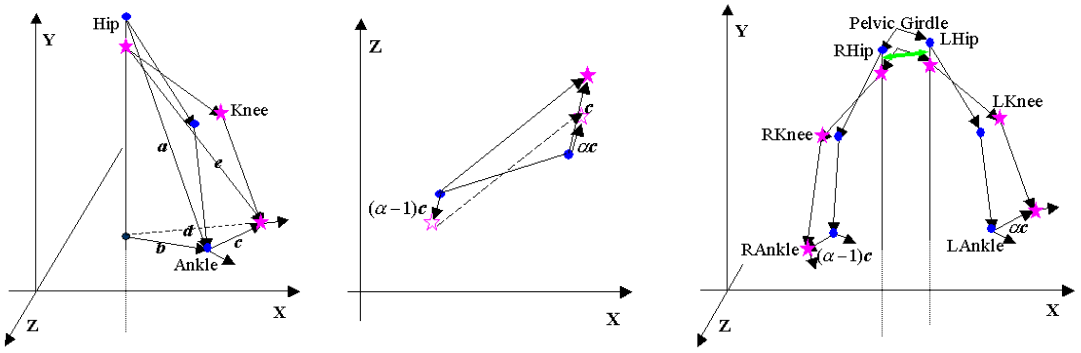
q is then concatenated with the quaternion computed from the original hip joint angles, so the ankle can be transformed to the new location. This completes the IK for a single leg.

However, stepping involves two legs. Say we want to shift the ankle of the leading leg by **c**. We cannot just ask one leg to try to make this shift because it will likely change the root's vertical position. The other leg will be affected by this root shift, and since its configuration is not changed, its ankle will be likely moved up into the air or down under the ground. So we need to distribute the shift **c** to both legs with two requirements:

1. The sum of the shifts from the two ankles equals **c**.

2. The vertical translation of the right hip equals that of the left hip, because both of them are connected to the pelvic girdle (see the green bi-directional arrow in Figure C.2(c)).

From requirement 1, we simply let the shift for the leading leg to be $\alpha\mathbf{c}$, and the shift of the stance leg to be $(\alpha - 1)\mathbf{c}$, see Figure C.2(b). Then by requirement 2, we equate the vertical shifts of the two hips. It happens that α has a simple closed-form solution for this hip constraint (see Appendix C.1).



(a) One-leg IK (b) Planar illustration (c) Two-leg IK

Figure C.2: IK algorithm illustration. Dots are original joint locations, stars are new joint locations.

Although we compute the pose difference by shifting both ankles, i.e., by shifting the dots to the imaginary hollow stars in Figure C.2(b), we do not really shift both ankles in the synthesized motion since the stance ankle has to remain fixed. The kinematic root is temporarily switched to the stance ankle after the leading leg takes off, and we perform a displacement mapping from the original hip rotations to the desired hip rotations during the swing phase of the leading leg, until its touchdown. The pose of the lower body is transformed so that we get a step from the left dot to

the solid star in Figure C.2(b). The imaginary dashed step was used for the IK pose computation, while the parallel solid step is the actual step that satisfies both the stance ankle constraint and the stepping ankle displacement requirement.

During the step transformation, two constraints have to be kept. First, due to the hip rotation shift, the stance ankle rotation has to be shifted in the reverse direction (opposite of $\mathbf{q}$) to counteract the hip rotation displacement and remain stable. Second, our IK basically keeps the knee flexion from the original stepping. However, a large ankle displacement $\mathbf{c}$ may need different knee flexion amplitudes to guarantee certain foot clearance for the swing foot (this can be seen from Figure C.1). We constantly check how close the swing foot is to the ground. When there is danger of the foot hitting the ground during the swing, the knee is flexed allowing the foot to follow the height of its original swing motion (see Appendix C.2). Figure 3.8(h) is an example of our IK. The original motion is a step to the right (see pressure data), and is modified to a right backward stepping by the IK to match the input pressure data.

C.1 Derivation of α

Denote the hip position as $\mathbf{h}$, the ankle position as $\mathbf{a}$. As illustrated in Figure C.2(a), the hip vertical translation t is calculated as follows

$$t = \|\mathbf{h}_y - \mathbf{a}_y\| - \sqrt{\|\mathbf{e}\|^2 - \|\mathbf{d}\|^2} \tag{C.4}$$

Equate the left hip translation with the right hip translation, we can get an equation of this form

$$\sqrt{a\alpha^2 + b_1\alpha + c_1} - \sqrt{a\alpha^2 + b_2\alpha + c_2} + d = 0 \tag{C.5}$$

The above equation can be deduced to a quadratic equation

$$A\alpha^2 + B\alpha + C = 0$$

We pick the solution in $[0, 1]$. In case the two solutions are both in this range, we pick the one closer to 0.5. There are cases when both solutions are out of the range $[0, 1]$, and the resulting animation is often weird. This is because the needed correction is too large (this often means a wrong match just happened), while the kinematics is satisfied, other constraints such as center of mass should stay in the feet support polygon is violated. In this case, we reduce the correction to be made in this step recursively until we get a reasonable solution for α, and the stepping error is distributed into IKs for the following steps as well, rather than trying to correct the error in one unrealistic step.

C.2 Flexion of Knee

Denote the orientation of the knee joint as $\mathbf{q} = (q_x, q_y, q_z, q_w)$, its inverse as $\mathbf{q}' = (-q_x, -q_y, -q_z, q_w)$, the ankle joint position in the knee joint coordinate frame as $\mathbf{s} = (s_x, s_y, s_z)$. In our case, $\mathbf{s} = (0, s_y, 0)$. From forward kinematics of the leg, we know we want to flex the knee to achieve an ankle height shift δy. Denote the transformation of a vector $\mathbf{s}$ by a quaternion $\mathbf{q}$ as:

$$\mathbf{q}(\mathbf{s}) = \mathbf{q} \otimes (s_x, s_y, s_z, 0) \otimes \mathbf{q}',$$

where $\otimes$ means quaternion multiplication. Denote the knee flexion angle we want as θ, then the quaternion corresponding to this flexion is $\mathbf{q}_\theta = (sin(\theta), 0, 0, cos(\theta))$, since the flexion axis is the $X = (1, 0, 0)$ for our skeleton model. From

$$Y[(\mathbf{q} \otimes \mathbf{q}_\theta)(\mathbf{s})] - Y[\mathbf{q}(\mathbf{s})] = \delta y, \tag{C.6}$$

where $Y(.)$ fetches the Y component of a vector. The above equation expands to

$$acos(\theta)^2 + bsin(\theta)cos(\theta) + c = 0, \tag{C.7}$$

where

$$\begin{aligned} a &= q_x^2 - q_y^2 + q_z^2 - q_w^2 \\ b &= 2(q_x q_w + q_y q_z) \\ c &= \frac{\delta y}{2 s_y} - a \end{aligned}$$

The solutions for Equation C.7 are:

$$\begin{aligned} \theta_1 &= arctan(\frac{4c\alpha - ad_1}{bd_1}) \\ \theta_2 &= arctan(\frac{4c\alpha - ad_2}{bd_2}) \end{aligned}$$

where

$$\begin{aligned} \alpha &= a^2 + b^2 \\ \beta &= 4ac - 2b^2 \\ \gamma &= 2\sqrt{b^4 - 4b^2ac - 4b^2c^2} \\ d1 &= \beta + \gamma \\ d2 &= \beta - \gamma \end{aligned}$$

Between (θ_1, θ_2), we choose the one with smaller amplitude as the solution. Note in our forward kinematics, $\mathbf{q}_{world} = \mathbf{q}_{parent\_world} \otimes \mathbf{q}_{local}$, the new knee local rotation should be $\mathbf{q}_{new\_local} = \mathbf{q}_{old\_local} \otimes \mathbf{q}_\theta$.

Appendix D

Details of Our Dynamics Simulator

Our rigid body simulator is based on a Lagrange multiplier approach for computing constraint forces, inspired by the work of [Baraff 1996]. We extend this approach by allowing the simulator to solve both forward and inverse dynamics problems. Contact and collision are handled as linear complementarity problem (LCP). We refer the readers to [Cline 2002; Cline et al. 2002] for more details.

D.1 Equations of Motion

We formulated the equations of motion as a full-coordinate constraint-based matrix form, instead of in reduced (generalized) coordinates. In the Lagrange multiplier approach, the velocity of each body is parameterized by a full six coordinate representation. Each joint in an articulated body is represented by a constraint equation, which is a linear equation on the velocities of the bodies. Constraint i between bodies a and b is given by the equation:

$$\mathsf{j}_{ia}\mathbf{v}_a + \mathsf{j}_{ib}\mathbf{v}_b = \mathbf{0}, \tag{D.1}$$

where the Jacobian matrices j have six columns and one row for each degree of freedom they remove from the system (the number of rows is referred to as the degree of the constraint). For a system with many constraints and many bodies we construct one large jacobian matrix J, containing all of the constraint equations, and concatenate the velocities of all of the bodies into a single vector $\mathbf{v}$. For example, if we had a chain of four rigid bodies connected by three joints, the constraint equation would appear as follows:

$$\mathsf{J}\mathbf{v} = \begin{bmatrix} \mathsf{j}_{1a} & \mathsf{j}_{1b} & 0 & 0 \\ 0 & \mathsf{j}_{2b} & \mathsf{j}_{2c} & 0 \\ 0 & 0 & \mathsf{j}_{3c} & \mathsf{j}_{3d} \end{bmatrix} \begin{bmatrix} \mathbf{v}_a \\ \mathbf{v}_b \\ \mathbf{v}_c \\ \mathbf{v}_d \end{bmatrix} = \mathbf{0} \tag{D.2}$$

If the constraint is workless (i.e., a frictionless joint), then the constraint forces are multiples of the rows of J. The sum of all constraint forces is given by

$$\mathbf{f}_c = \mathsf{J}^T \lambda, \tag{D.3}$$

where λ is a vector of Lagrange multipliers.

The row space of J is the space of constraint forces. We now wish to introduce an analogous matrix H whose row space is the space of all possible "muscle forces", or more precisely "joint torques", which the muscles surrounding a joint can apply to their neighboring bodies. The rows of H correspond to equal and opposite torques applied at the joint. Similar to equation D.3, the muscle forces are given by

$$\mathbf{f}_m = \mathsf{H}^T \tau. \tag{D.4}$$

Combining the constraint equations with the Newton-Euler equations of motion gives us the following matrix equation:

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T & -\mathsf{H}^T \\ \mathsf{J} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \lambda \\ \tau \end{bmatrix} = \begin{bmatrix} \mathbf{f}_x \\ 0 \end{bmatrix} \tag{D.5}$$

Here M is the mass-inertia matrix of the bodies in the system (a block diagonal matrix with each block corresponding to one body), $\mathbf{a}$ is the acceleration vector of the bodies, and vector $\mathbf{f}_x$ contains external forces such as gravity and perturbation forces [1].

Discretize Equation D.5, we have

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T & -\mathsf{H}^T \\ \mathsf{J} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ h\lambda \\ h\tau \end{bmatrix} = \begin{bmatrix} \mathsf{M}\mathbf{v}_t + h\mathbf{f}_x \\ \mathbf{c}_e \end{bmatrix} \tag{D.6}$$

where h is the time step size, $\mathbf{v}_t$ and $\mathbf{v}_{t+h}$ are the velocity of the rigid bodies at time t and $t+h$. To counteract drift at the joints due to numerical error, we use a Baumgarte stabilization scheme [Baumgarte 1972]. $\mathbf{c}_e$ stands for the equality constraint stabilization quantity. The Post-step stabilization scheme described in [Ascher et al. 1995; Cline and Pai 2003] is another option, where after each simulation step we project the position of the bodies onto the constraint manifold. Post-step stabilization is slower than Baumgarte, but is more accurate and stable.

Equation D.6 is a unified expression that is true for both forward and inverse dynamics. We will later rearrange this equation to reflect the known and unknown values in these two types of problems.

[1] We also put negative angular bias force into external forces to simplify the left hand side matrix. The angular bias force $[\omega]\mathsf{I}\omega$ is the force which must be applied to a rigid body to produce zero angular acceleration.

D.2 Details of Matrices J and H

For a joint connecting one body to another, the matrices J and H both have twelve columns (they multiply with a vector $[\nu_a^T \omega_a^T \nu_b^T \omega_b^T]^T$ containing the linear and angular velocities of both bodies that the joint is connected to). The number of rows in J is the number of degrees of freedom (DoF) that the joint removes from the system, while the number of rows of H is the number of degrees of freedom in the rotation of the joint. These two numbers always add up to six. For instance, in a hinge joint, there is one degree of rotation freedom, and five DoFs are removed from the system. For a ball joint, there are three degrees of rotational freedom, and three DoFs are removed. If we recall that the rows of these matrices are used as force basis vectors, we can also note that the first and second halves of these vectors must correspond to equal and opposite forces applied to the pair of bodies. This constraint restricts the row space of J and H to a six-dimensional subspace of $\mathbb{R}^{12}$. The row space J and H must always span this entire 6D subspace containing all equal-and-opposite force pairs.

In our implementation, we follow the convention for describing rigid body velocities that is described by Baraff and Witkin [1997]. The velocity of a rigid body is given as a vector $\mathbf{v} = [\nu^T \omega^T]^T$, where ν^T is the velocity of the centre of mass of the object, given in world coordinates, and ω is the world coordinates of the angular velocity vector. Under this convention, our matrices J and H has the following structure:

$$\begin{bmatrix} \mathsf{J} \\ \mathsf{H} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & & -1 & 0 & 0 & \\ 0 & 1 & 0 & -[\mathbf{r}_a] & 0 & -1 & 0 & [\mathbf{r}_b] \\ 0 & 0 & 1 & & 0 & 0 & -1 & \\ 0 & 0 & 0 & e_1^T & 0 & 0 & 0 & -e_1^T \\ 0 & 0 & 0 & e_2^T & 0 & 0 & 0 & -e_2^T \\ 0 & 0 & 0 & e_3^T & 0 & 0 & 0 & -e_3^T \end{bmatrix}, \tag{D.7}$$

where $\mathbf{r}_a$ is the vector from the centre of mass of body A to the joint, and $\mathbf{r}_b$ is the vector from body B's centre of mass to the joint. The notation $[\mathbf{r}]$ denotes the 3×3 skew-symmetric cross product matrix of vector $\mathbf{r}$. The axis vectors e_1, e_2, e_3 are three orthogonal vectors in world coordinates, some of which are the free rotation axes of the joint, and some of which may be axes that joint bodies are constrained not to rotate around. Which of the rows of the above matrix belong to J and which belong to H depends on the type of the joint.

D.3 Forward Dynamics

In forward dynamics with known control, the muscle force multipliers τ are known quantities. Moving $\mathsf{H}^T\tau$ to the right hand side of Equation D.6 gives

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T \\ \mathsf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ h\lambda \end{bmatrix} = \begin{bmatrix} \mathsf{M}\mathbf{v}_t + h\mathbf{k} \\ \mathbf{c}_e \end{bmatrix} \tag{D.8}$$

where $\mathbf{k} = \mathbf{f}_x + \mathsf{H}^T\tau$. Solving this equation at each time step gives us the updated velocity of the system.

D.3.1 Accommodating Stiffness

In order for the techniques to work for large time steps and achieve real-time simulation, an implicit integrator is essential. We use an implementation of the linearly implicit time stepping method as described by Anitescu and Potra [2002]. Fortunately this requires only small modifications to an explicit integrator. The main requirement of this implicit method is that we must calculate the gradients of the stiff forces with respect to changes in the position and velocity of the rigid bodies.

To derive the linearly implicit method, we start with a backward Euler discretization of rigid body dynamics equation:

$$\mathsf{M}\Big(\frac{\mathbf{v}_{t+h} - \mathbf{v}_t}{h}\Big) = \mathbf{k}\big(\mathbf{p}_{t+h}, \mathbf{v}_{t+h}, t+h\big) \tag{D.9}$$

The difficulty in solving this is that the force vector $\mathbf{k}$ is not known unless the position and velocity vectors $\mathbf{p}_{t+h}$ and $\mathbf{v}_{t+h}$ are known. We make the linear approximation (hence the term "linearly implicit") that

$$\begin{aligned}\mathbf{k}\big(\mathbf{p}_{t+h}, \mathbf{v}_{t+h}, t+h\big) \approx \\ \mathbf{k}\big(\mathbf{p}_t, \mathbf{v}_t, t\big) + \nabla_p \mathbf{k} h\ \mathbf{v}_{t+h} + \nabla_v \mathbf{k}(\mathbf{v}_{t+h} - \mathbf{v}_t),\end{aligned} \tag{D.10}$$

where $\nabla_p \mathbf{k}$ and $\nabla_v \mathbf{k}$ are the gradients of the function $\mathbf{k}$ with respect to change in position and velocity, respectively, and evaluated at $(\mathbf{p}_t, \mathbf{v}_t, t)$.

If we substitute this into equation D.9 and move all of the terms with $\mathbf{v}_{t+h}$ to the left hand side, we obtain

$$\begin{aligned}(\mathsf{M} - h^2 \nabla_p \mathbf{k} - h \nabla_v \mathbf{k})\mathbf{v}_{t+h} = \\ \mathsf{M}\mathbf{v}_t - h \nabla_v \mathbf{k} \mathbf{v}_t + h\ \mathbf{k}\big(\mathbf{p}_t, \mathbf{v}_t, t\big)\end{aligned} \tag{D.11}$$

It is convenient to use the notation

$$\hat{\mathsf{M}} = \mathsf{M} - h^2 \nabla_p \mathbf{k} - h \nabla_v \mathbf{k} \tag{D.12}$$

and

$$\hat{\mathbf{k}} = \mathbf{k}\big(\mathbf{p}_t, \mathbf{v}_t, t\big) - \nabla_v \mathbf{k} \mathbf{v}_t \tag{D.13}$$

so that the linearly implicit equation for forward dynamics closely resembles Equation D.8. The implicit version is:

$$\begin{bmatrix} \hat{\mathsf{M}} & -\mathsf{J}^T \\ \mathsf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ h\lambda \end{bmatrix} = \begin{bmatrix} \mathsf{M}\mathbf{v}_t + h\hat{\mathbf{k}} \\ \mathbf{c}_e \end{bmatrix} \tag{D.14}$$

In our implementation, we calculate the force gradients $\nabla_p \mathbf{k}$ and $\nabla_v \mathbf{k}$ using automatic differentiation techniques [Nocedal and Wright 1999].

Bibliography

Abdallah, M., and Goswami, A. 2005. A biomechanically motivated two-phase strategy for biped upright balance control. In *International Conference on Robotics and Automation*, 2008–2013.

Abe, Y., Liu, C. K., and Popovic, Z. 2004. Momentum-based parameterization of dynamic character motion. In *Symposium on Computer Animation 2004*, 173–182.

Adrian, M. J., and Cooper, J. M. 1989. *Biomechanics of Human Movement*. Benchmark Press, Inc.

Alexander, R. M. 1992. *The Human Machine*. Columbia University Press.

Anitescu, M., and Potra, F. A. 2002. A time-stepping method for stiff multibody dynamics with contact and friction. *International Journal for Numerical Methods in Engineering 55*, 753–784.

Arikan, O., and Forsyth, D. A. 2003. Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3, 483–490.

Arikan, O., Forsyth, D. A., and O'Brien, J. F. 2003. Motion synthesis from annotations. *ACM Transactions on Graphics 22*, 3, 402–408.

Arikan, O., Forsyth, D. A., and O'Brien, J. F. 2005. Pushing people around. In *Symposium on Computer Animation*.

Ascher, U. M., Chin, H., Petzold, L. R., and Reich, S. 1995. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Mechanics of Structures and Machines 23*, 135–158.

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. *ACM Trans. on Graphics 14*, 4, 311–336.

Baraff, D., and Witkin, A. 1997. *Physically Based Modeling: Principles and Practice*. Siggraph '97 Course notes.

Baraff, D. 1996. Linear-time dynamics using lagrange multipliers. In *Proceedings of SIGGRAPH 1996*, H. Rushmeier, Ed., 137–146.

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

Belkasim, S. O., Shridhar, M., and Ahmadi, M. 1991. Pattern recognition with moment invariants: a comparative study. *Pattern Recognition 24*, 12, 1117–1138.

Belland, C., Davis, J. W., Helfer, B., Varadarajan, S., Gleicher, M., and King, S. 2002. *Motion Capture: Pipeline, Applications, and Use*.

Bodenheimer, B., Rose, C., Rosenthal, S., and Pella, J. 1997. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97*, Eurographics, 3–18.

Bottino, A., and Laurentini, A. 2001. Experimenting with nonintrusive motion capture in a virtual environment. *The Visual Computer 17*, 1, 14–29.

Bruderlin, A., and Williams, L. 1995. Motion signal processing. In *SIGGRAPH 95 Conference Proceedings*, 97–104.

Buchanan, J. J., and Horak, F. B. 2001. Transitions in a postural task: do the recruitment and suppression of degrees of freedom stabilize posture? *Experimental Brain Research 139*, 482–494.

Buhler, M., and Koditschek, D. E. 1988. Analysis of a simplified hopping robot. In *International Conference on Robotics and Automation*, 817–819.

Cavazza, M., Earnshaw, R., Magnenat-Thalmann, N., and Thalmann, D. 1998. Survey: Motion control of virtual humans. *IEEE Computer Graphics and Applications 18*, 5, 24–31.

Chai, J., and Hodgins, J. K. 2005. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (August).

Chang, K. S., and Khatib, O. 2000. Operational space dynamics: Efficient algorithms for modeling and control of branching mechanisms. In *IEEE Intern. Conf. on Robotics and Automation*, 850–856.

Chow, D. H., Holmes, A. D., and Tse, A. T. 2002. Sudden release during a pulling task: the effect of release load on stance perturbation and recovery. *Gait Posture 15*, 3, 266–273.

Chua, P. T., Crivella, R., Daly, B., Hu, N., Schaaf, R., Ventura, D., Camill, T., Hodgins, J. K., and Pausch, R. 2003. Training for physical tasks in virtual environments: Tai chi. In *Proceedings of IEEE Virtual Reality 2003*, IEEE, 87–96.

Cline, M. B., and Pai, D. K. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Cline, M. B., Yin, K., and Pai, D. K. 2002. Motion perturbation based on simple neuromotor control models. Tech. Rep. TR-2002-03.

Cline, M. B. 2002. *Rigid Body Simulation with Contact and Constraints*. Master's thesis, University of British Columbia.

Cohen, M. F. 1992. Interactive spacetime control for animation. In *Proceedings of*

SIGGRAPH 1992, 293–302.

Craig, J. J. 1989. *Introduction to Robotics: Mechanics and Control*, second ed. Addison-Wesley Publishing Company.

Crowe, A., Porrill, J., and Prescott, T. 1998. Kinematic coordination of reach and balance. *Journal of Motor Behavior 30*, 3, 217–233.

Dasgupta, A., and Nakamura, Y. 1999. Making feasible walking motion of humanoid robots from human motion capture data. In *Robotics and Automation*, vol. 2, 1044–1049.

DasGupta, A., Nakamura, Y., and Yoshimoto, K. 1998. Analysis and synthesis of motion for a humanoid robot using human motion data. In *2nd Japan-China Bilateral Symposium on Advanced Manufacturing Engineering*.

Eng, J. J., Winter, D. A., and Patla, A. E. 1994. Strategies for recovery from a trip in early and late swing during human walking. *Experimental Brain Research 102*, 2, 339–349.

Eng, J. J., Winter, D. A., and Patla, A. E. 1997. Intralimb dynamics simplify reactive control strategies during locomotion. *Journal of Biomechanics 30*, 6, 581–588.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, 251–260.

Fang, A. C., and Pollard, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3, 417–426.

Featherstone, R. 1987. *Robot Dynamics Algorithms*. Kluwer.

Flash, T., and Hogan, N. 1985. The coordination of arm movements: an experimentally confirmed mathematical model. *J Neurosci 5*, 1688–1703.

Ghahramani, Z., and Hinton, G. E. 1996. The em algorithm for mixtures of factor analyzers. Tech. Rep. CRG-TR-96-1, Department of Computer Science.

Gleicher, M. 1998. Retargetting motion to new characters. In *Proceedings of SIGGRAPH 1998*, 33–42.

Goldfarb, L. W. 1993. *Mind in Motion: Why Robots Fall Down*. `http://www.mindinmotion-online.com/robots.html/`.

Gomi, H., and Kawato, M. 1997. Human arm stiffness and equilibrium-point trajectory during multi-joint movement. *Biological Cybernetics 76*, 163–171.

Goswami, A., and Kallem, V. 2004. Rate of change of angular momentum and balance maintenance of biped robots. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 3785–3790.

Goswami, A. 1999. Postural stability of biped robots and the foot rotation indicator (fri) point. *International Jounal of Robotics Research 18*, 6, 523–533.

Grassia, F. S. 1998. Practical parameterization of rotations using the exponential map. *The Journal of Graphics Tools 3*, 3, 29–48.

Grochow, K., Martin, S. L., Hertzmann, A., and Popovic, Z. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3, 522–531.

Grzeszczuk, R., and Terzopoulos, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 1995*, 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 1998*, 9–20.

Hall, S. J. 1998. *Basic Biomechanics*, second ed. Mosby.

Harada, T., Sato, T., and Mori, T. 2001. Pressure distribution image based human motion tracking using skelton and surface integration model. In *IEEE Intern. Conf. on Robotics and Automation*, vol. 5, 3201–3207.

Harada, K., Kajita, S., Kaneko, K., and Hirukawa, H. 2003. ZMP analysis for arm/leg coordination. In *IEEE Intern. Conf. on Intelligent Robots and Systems*, IEEE, 75–81.

Harris, C. M., and Wolpert, D. M. 1998. Signal-dependent noise determines motor planning. *Nature 394*, 6695, 780–784.

Harrison, J., Rensink, R. A., and van de Panne, M. 2004. Obscuring length changes during animated motion. *ACM Trans. Graph. 23*, 3, 569–573.

Hastie, T., Tibshirani, R., and Friedman, J. 2001. *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer-Verlag.

Henry, S. M., Fung, J., and Horak, F. B. 1998. Emg responses to maintain stance during multidirectional surface translations. *Journal of Neurophysiology 80*, 4, 1939–1950.

Hertzmann, A. 2003. Machine learning for computer graphics: A manifesto and tutorial. In *The Eleventh Pacific Conference on Computer Graphics and Applications*, 22–36.

Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. 1998. The development of honda humanoid robot. In *IEEE International Conference on Robotics & Automation*, 1321–1326.

Hirschfeld, H., and Forssberg, H. 1991. Phase-dependent modulations of anticipatory postural activity during human locomotion. *Journal of Neurophysiology 66*, 1, 12–19.

Hirukawa, H., Kajita, S., Kanehiro, F., Kaneko, K., and Isozumi, T. 2003. The human-size humanoid robot that can walk, lie down and get up. In *11th International Symposium of Robotics Research*.

Hodgins, J. K., and Pollard, N. S. 1997. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH '97*, 153–162.

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

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

Hodgins, J. K. 1991. Biped gait transitions. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Hogan, N. 1990. Mechanical impedance of single and multi-articular systems. In *Multiple Muscle Systems: Biomechanics and Movement Organization*, J. Winters and S. Woo, Eds. Springer-Verlag, ch. 9, 149–64.

Honda Motor Co., L., 2006. Studies of leg/foot functions of the robot, `http://world.honda.com/asimo/p3/technology/`.

Hoogvliet, P., van Duyl, W. A., de Bakker, J. V., Mulder, P. G. H., and Stam, H. J. 1997. A model for the relation between the displacement of the ankle and the center of pressure in the frontal plane, during one-leg stance. *Gait Posture 6*, 1, 39–49.

Horak, F. B., Henry, S. M., and Shumway-Cook, A. 1997. Postural perturbations: new insights for treatment of balance disorders. *Physical Therapy 77*, 5, 517–534.

Hornby, G. S., Fujita, M., Takamura, S., Yamamoto, T., and Hanagata, O. 1999. Autonomous evolution of gaits with the Sony quadruped robot. In *Generic and Evolutionary Computation Conference*, 1305–1312.

Hsiao, E. T., and Robinovitch, S. N. 1998. Common protective movements govern unexpected falls from standing height. *Journal of Biomechanics 31*, 1, 1–9.

Hsiao, E. T., and Robinovitch, S. N. 1999. Biomechanical influences on balance recovery by stepping. *Journal of Biomechanics 32*, 10, 1099–1106.

Huang, Q., and Nakamura, Y. 2005. Sensory reflex control for humanoid walking. *IEEE Trans. Robotics 21*, 5, 977–984.

Ito, S., and Kawasaki, H. 2000. A standing posture control based on ground reaction force. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Ito, S., Asano, H., and Kawasaki, H. 2003. A balance control in biped double support phase based on center of pressure of ground reaction forces. In *7th IFAC Symposium on Robot Control*, 205–210.

Jenkins, O., and Mataric, M. 2003. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Jordan, M. I., and Wolpert, D. M. 1999. Computational motor control. In *The Cognitive Neurosciences*, M. Gazzaniga, Ed. MIT Press.

Kagami, S., Kanehiro, F., Tamiya, Y., Inaba, M., and Inoue, H. 2000. Autobalancer: An online dynamic balance compensation scheme for humanoid robots. In *Fourth Int. Workshop on Algorithmic Foundations on Robotics*.

Kagami, S., Mochimaru, M., Ehara, Y., Miyata, N., Nishiwaki, K., Kanade, T., and Inoue, H. 2003. Measurement and comparison of humanoid h7 walking with human being. In *International Conference on Humanoid Robotics*.

Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. 2001. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *IEEE Intern. Conf. on Intelligent Robots and Systems*, IEEE, 239–246.

Kajita, S., Matsumoto, O., and Saigo, M. 2001. Real-time 3d walking pattern generation for a biped robot with telescopic legs. In *International Conference on Robotics & Automation*, IEEE, 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 *International Conference on Robotics and Automation*.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. 2003. Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In *IEEE Intern. Conf. on Intelligent Robots and Systems*, IEEE, 1644–1650.

Kandel, E. R., Schwartz, J. H., and Jessell, T. M. 2000. *Principles Of Neural Science*, fourth ed. McGraw–Hill.

Kanehiro, F., Hirukawa, H., and Kajita, S. 2004. OpenHRP: Open architecture humanoid robotics platform. *The International Journal of Robotics Research 23*, 2, 155–165.

Kaneko, K., Kanehiro, F., Kajita, S., Yokoyama, K., Akachi, K., Kawasaki, T., Ota, S., and Isozumi, T. 2002. Design of prototype humanoid robotics platform for hrp. *IEEE/RSJ Intl. Conf. on Intell. Robots and Systems*, 2431–2436.

Kawato, M., Furukawa, K., and Suzuki, R. 1987. A hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics 57*, 3, 169–185.

Kawato, M. 1999. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology 9*, 718–727.

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, 1–15.

Kim, J., Chung, W. K., Youm, Y., and Lee, B. H. 2002. Real-time ZMP compensation method using null motion for mobile manipulators. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 2400–2405.

Kim, J., Park, I., and Oh, J. 2006. Experimental realization of dynamic walking of biped humanoid robot KHR-2 using ZMP feedback and inertial measurement.

Advanced Robotics 20, 6 (June), 707 – 736.

Komura, T., and Shinagawa, Y. 1997. A muscle-based feed-forward controller of the human body. In *Computer Graphics Forum (Proceedings of Eurographics 1997)*, 165–176.

Komura, T., Shinagawa, Y., and Kunii, T. 2001. Attaching physiological effects to motion-captured data. In *Graphics Interface Proceedings*, 27–36.

Komura, T., Ho, E. S. L., and H.Lau, R. W. 2005. Animating reactive motion using momentum-based inverse kinematics. *Computer Animation and Virtual Worlds 16*, 3-4, 213–223.

Komura, T., Leung, H., Kudoh, S., and Kuffner, J. 2005. A feedback controller for biped humanoids that can counteract large perturbations during gait. In *International Conference on Robotics and Automation*, 2001–2007.

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. In *SIGGRAPH 2002 Conference Proceedings*, 473–482.

Kovar, L., Gleicher, M., and Schreiner, J. 2002. Footskate cleanup for motion capture editing. In *ACM SIGGRAPH Symposium on Computer Animation*, 97–104.

Kudoh, S., and Komura, T. 2003. C^2 continuous gait-pattern generation for biped robots. In *International Conference on Intelligent Robots and Systems*.

Kudoh, S., Komura, T., and Ikeuchi, K. 2002. The dynamic postural adjustment with the quadratic programming method. In *International Conference on Intelligent Robots and Systems*.

Kudoh, S., Komura, T., and Ikeuchi, K. 2006. Stepping motion for a human-like character to maintain balance against large perturbations. In *International Conference on Intelligent Robotics and Automation*, 2661–2666.

Kunivoshi, Y., Ohmura, Y., Terada, K., Yamamoto, T., and Nagakubo, A. 2003. Exploiting the global dynamics structure of whole-body humanoid motion - getting the "knack"of roll-and-rise motion. In *11th International Symposium of Robotics Research*.

Kuo, A. D., and Zajac, F. E. 1993. Human standing posture: multi-joint movement strategies based on biomechanical constraints. *Progress in Brain Research 97*, 349–358.

Kuo, A. D. 1999. Stabilization of lateral motion in passive dynamic walking. *The International Journal of Robotics Research 18*, 917–930.

Laszlo, J. F., van de Panne, M., and Fiume, E. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of ACM SIGGRAPH*, ACM Press, 155–162.

Laszlo, J. 1996. *Controlling Bipedal Locomotion for Computer Animation*. Master's thesis, University of Toronto.

Latash, M., and Zatsiorsky, V. 1993. Joint stiffness: Myth or reality. *Human*

Movement Science 12, 653–692.

Lawrence, N. D. 2004. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems 16*.

Lee, S., and Goswami, A. 2007. Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 0–0.

Lee, J., and Shin, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 1999*, 39–48.

Lee, S., and Terzopoulos, D. 2006. Heads up!: biomechanical modeling and neuromuscular control of the neck. *ACM Transactions on Graphics 25*, 3, 1188–1198.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3, 491–500.

Lee, D. D. 2003. Learning in sensorimotor systems. In *Advances in Neural Information Processing Systems 16: Tutorial*.

Li, Y., Wang, T., and Shum, H. Y. 2002. Motion texture: A two-level statistical model for character motion synthesis. In *SIGGRAPH 2002 Conference Proceedings*, 465–471.

Liu, C. K., and Popovic, Z. 2002. Synthesis of complex dynamic character motion from simple animations. In *Proceedings of SIGGRAPH 2002*, 408–416.

Liu, Z., Gortler, S. J., and Cohen, M. F. 1994. Hierarchical spacetime control. In *Proceedings of SIGGRAPH 1994*, 24–29.

Liu, C. K., Hertzmann, A., and Popovi, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3, 1071–1081.

Maki, B. E., and McIlroy, W. E. 1999. The control of foot placement during compensatory stepping reactions: does speed of response take precedence over stability? *IEEE Transations on Rehabilitation Engineering 7*, 1, 80–90.

Maki, B. E., McIlroy, W. E., and Perry, S. D. 1996. Influence of lateral destabilization on compensatory stepping responses. *J Biomech 29*, 3, 343–353.

Maki, B. E., McIlroy, W. E., and Fernie, G. R. 2003. Change-in-support reactions for balance recovery. *IEEE Engineering in Medicine and Biology Magazine 22*, 2, 20–26.

Marigold, D. S., and Patla, A. E. 2001. Strategies for dynamic stability during locomotion on a slippery surface: effects of prior experience and knowledge. *Journal of Neurophysiology 88*, 1, 339–353.

Marigold, D. S., Bethune, A. J., and Patla, A. E. 2003. Role of the unperturbed limb and arms in the reactive recovery response to an unexpected slip

during locomotion. *Journal of Neurophysiology 89*, 4, 1727–1737.

Mason, M. T. 2001. *Mechanics of Robotic Manipulation*. MIT Press.

Matusik, W., Pfister, H., Brand, M., and McMillan, L. 2003. A data-driven reflectance model. *ACM Transactions on Graphics 22*, 3, 759–769.

McIlroy, W. E., and Maki, B. E. 1999. The control of lateral stability during rapid stepping reactions evoked by antero-posterior perturbation: does anticipatory control play a role? *Gait Posture 9*, 3, 190–198.

Medved, V. 2001. *Measurement of Human Locomotion*. CRC Press.

Miura, H., and Shimoyama, I. 1984. Dynamic Walk of a Biped. *The International Journal of Robotics Research 3*, 2, 60.

Moeslund, T. B., and Granum, E. 2001. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding: CVIU 81*, 3, 231–268.

Moore, K. L., and Dalley, A. F. 1999. *Clinically Oriented Anatomy*, fourth ed. Lippincott Williams & Wilkins.

Morimoto, J., and Atkeson, C. G. 2003. Minimax differential dynamic programming: An application to robust biped walking. In *NIPS 15*, 1539–1546.

Morimoto, J., Cheng, G., Atkeson, C. G., and Zeglin, G. 2004. A simple reinforcement learning algorithm for biped walking. In *Proc. IEEE Int'l Conf. on Robotics and Automation*.

Morimoto, J., Nakanishi, J., Endo, G., Cheng, G., Atkeson, C., and Zeglin, G. 2005. Poincaré-map-based reinforcement learning for biped walking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*.

Murray, R. M., Li, Z., and Sastry, S. S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

Mussa-Ivaldi, F. A. 1999. Modular features of motor control and learning. *Current Opinion in Neurobiology 9*, 713–717.

Nakanishi, J., and Schaal, S. 2004. Feedback error learning and nonlinear adaptive control. *Neural Networks 17*, 1453–1465.

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., and Kawato, M. 2003. Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives. In *Workshop on Robot Learning by Demonstration, IEEE Int'l Conf. Intelligent Robots and Systems*.

Nakaoka, S., Nakazawa, A., Yokoi, K., Hirukawa, H., and Ikeuchi, K. 2003. Generating whole body motions for a biped humanoid robot from captured human dances. In *International Conference on Robotics and Automation*.

NaturalMotion. NaturalMotion Ltd. `http://www.naturalmotion.com`.

Ngo, J. T., and Marks, J. 1993. Spacetime constraints revisited. In *Proceedings of SIGGRAPH 1993*, 343–350.

Nigg, B. M., and Herzog, W. 1995. *Biomechanics of the Musculo-Skeletal System*. John Wiley & Sons Ltd.

Nocedal, J., and Wright, S. J. 1999. *Numerical Optimization*. Springer.

ODE. Open dynamics engine. `http://www.ode.org`.

Okumura, Y., Tawara, T., Endo, K., Furuta, T., and Shimizu, M. 2003. Realtime ZMP compensation for biped walking robot using adaptive inertia force control. In *International Conference on intelligent Robots and Systems*.

Oore, S., Terzopoulos, D., and Hinton, G. 2002. A desktop input device and interface for interactive 3D character animation. In *Graphics Interface 2002 Conference Proceedings*, 133–140.

Pai, Y. C., and Patton, J. 1997. Center of mass velocity-position predictions for balance control. *Journal of Biomechanics 30*, 4, 347–354.

Pai, Y. C., Maki, B. E., Iqbal, K., McIlroy, W. E., and Perry, S. D. 2000. Thresholds for step initiation induced by support-surface translation: a dynamic center-of-mass model provides much better prediction than a static model. *Journal of Biomechanics 33*, 3, 387–392.

Pai, D. K., Sueda, S., and Wei, Q. 2005. Fast physically based musculoskeletal simulation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, 25.

Pai, D. K. 1990. Programming anthropoid walking: Control and simulation. Tech. Rep. Computer Science Tech Report TR 90-1178, Cornell University.

Pai, D. K. 1991. Least constraint: A framework for the control of complex mechanical systems. In *Proceedings of the American Control Conference*, 1615–1621.

Parent, R. 2002. *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann.

Park, J., Chung, W. K., and Youm, Y. 1996. Weighted decomposition of kinematics and dynamics of kinematically redundant manipulators. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 480–486.

Park, M. J., Choi, M. G., and Shin, S. Y. 2002. Human motion reconstruction from inter-frame feature correspondences of a single video stream using a motion library. In *ACM SIGGRAPH Symposium on Computer Animation*, 113–120.

Park, S. I., Shin, H. J., and Shin, S. Y. 2002. On-line locomotion generation based on motion blending. In *ACM SIGGRAPH Symposium on Computer Animation*, 105–112.

Patla, A. E., Adkin, A., and Ballard, T. 1999. Online steering: coordination and control of body center of mass, head and body reorientation. *Experimental Brain Research 129*, 4, 629–634.

Patla, A. E. 2003. Strategies for dynamic stability during adaptive human locomotion. *IEEE Engineering in Medicine and Biology Magazine 22*, 2, 48–52.

Patton, J. L. 1998. *Global Modeling of Adaptive, Dynamic Balance Control*. PhD thesis, Northwestern University.

Perreault, E. J., Crago, P. E., and Kirsch, R. F. 2000. Estimation of intrinsic and reflex contributions to muscle dynamics: a modeling study. *IEEE Transactions on Biomedical Engineering 47*, 11, 1413–1421.

Peterka, R. J. 2003. Simplifying the complexities of maintaining balance. *IEEE Engineering in Medicine and Biology Magazine 22*, 2, 63–68.

Phillips, P. J., and Newton, E. M. 2002. Meta-analysis of face recognition algorithms. In *Proceedings of IEEE Automatic Face and Gesture Recognition*, 235–241.

Pollard, N. S., and Behmaram-Mosavat, F. 2000. Force-based motion editing for locomotion tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Popovic, Z., and Witkin, A. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH 1999*, 11–20.

Popovic, M., Hofmann, A., and Herr, H. 2004. Angular momentum regulation during human walking: Biomechanics and control. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 2405–2411.

Pratt, J., Carff, J., Drakunov, S., and Goswami, A. 2006. Capture point: A step toward humanoid push recovery. In *IEEE-RAS International Conference on Humanoid Robots*.

Raibert, M. H., and Hodgins, J. K. 1991. Animation of dynamic legged locomotion. In *Proceedings of SIGGRAPH '91*, 349–358.

Raibert, M. H. 1986. *Legged Robots That Balance*. MIT Press.

Reitsma, P. S. A., and Pollard, N. S. 2003. Perceptual metrics for character animation: Sensitivity to errors in ballistic motion. In *SIGGRAPH 2003 Conference Proceedings*.

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

Rietdyk, S., and Patla, A. E. 1998. Context-dependent reflex control: some insights into the role of balance. *Experimental Brain Research 119*, 2, 251–259.

Rietdyk, S., Patla, A. E., Winter, D. A., Ishac, M. G., and Little, C. E. 1999. Balance recovery from medio-lateral perturbations of the upper body during standing. *Journal of Biomechanics 32*, 11, 1149–1158.

Rose, J., and Gamble, J. G. 1994. *Human Walking*, second ed. Williams & Wilkins.

Rose, J., and Gamble, J. G. 2006. *Human Walking*, third ed. Williams & Wilkins.

Rose, C., Guenter, B., Bodenheimer, B., and Cohen, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 1996*, 147–154.

Rose, C., Cohen, M. F., and Bodenheimer, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–41.

Rose, C. F., Sloan, P. J., and Cohen, M. F. 2001. Artist-directed inverse-kinematics using radial basis function interpolation. *Comput. Graph. Forum 20*, 3.

Rosenbaum, D. A., Meulenbroek, R. J., and Vaughan, J. 1999. Remembered positions: stored locations or stored postures? *Experimental Brain Research 124*, 503–512.

Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 22, 2323–2326.

Roweis, S., Saul, L. K., and Hinton, G. E. 2002. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14*.

Runge, C. F., Shupert, C. L., Horak, F. B., and Zajac, F. E. 1999. Ankle and hip postural strategies defined by joint torques. *Gait and Posture 10*, 161–170.

Safonova, A., Pollard, N., and Hodgins, J. K. 2003. Optimizing human motion for the control of a humanoid robot. In *2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM2003)*.

Safonova, A., Hodgins, J. K., and Pollard, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph. 23*, 3, 514–521.

Salesin, D. 2003. The need for machine learning in computer graphics. In *Invited Talk in Advances in Neural Information Processing Systems 15*.

Santello, M., Flanders, M., and Soechting, J. F. 1998. Postural hand synergies for tool use. *The Journal of Neuraoscience 18*, 23, 10105–10115.

Schilling, R. J. 1990. *Fundamentals of Robotics: Analysis and Control*. Prentice Hall.

Schillings, A. M., van Vezel, B. M. H., Mulder, T. H., and Duysens, J. 1999. Widespread short-latency stretch reflexes and their modulation during stumbling over obstacles. *Brain Research 816*, 2, 480–486.

Schillings, A. M., van Vezel, B. M. H., Mulder, T. H., and Duysens, J. 2000. Muscular responses and movement strategies during stumbling over obstacles. *Journal of Neurophysiology 83*, 4, 2093–2102.

Sharon, D., and van de Panne, M. 2005. Synthesis of controllers for stylized planar bipedal walking. In *Intern. Conf. on Robotics and Automation*, IEEE, 2398–2403.

Shin, H., Kovar, L., and Gleicher, M. 2003. Physical touch-up of human motions. In *The Eleventh Pacific Conference on Computer Graphics and Applications*, 194–203.

Shumway-Cook, A., and Woollacott, M. H. 2001. *Motor Control: theory and*

practical applications, second ed. Lippincott Williams & Wilkins.

Sims, K. 1994. Evolving virtual creatures. In *Proceedings of SIGGRAPH 1994*, 15–22.

Smith, R. L. 1998. *Intelligent Motion Control with an Artificial Cerebellum*. PhD thesis, University of Auckland.

Sok, K., Kim, M., and Lee, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph. 26*, 3, 0–0.

Strogatz, S. H. 1994. *Nonlinear dynamics and Chaos : with applications to physics, biology, chemistry, and engineering*.

Sugihara, T., and Nakamura, Y. 2002. Whole-body cooperative balancing of humanoid robot using cog jacobian. In *Proc. of Intl Conf. on Intelligent Robots and Systems*, 2575–2580.

Taga, G., Yamaguchi, Y., and Shimizu, H. 1991. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics 65*, 147–159.

Tak, S., Song, O.-Y., and Ko, H.-S. 2000. Motion balance filtering. In *Computer Graphics Forum (Eurographics 2000)*, vol. 19(3), 437–446.

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. *J. Neurophysiology 86* (Aug).

Tedrake, R., and Seung, H. S. 2002. Improved dynamic stability using reinforcement learning. In *International Conference on Climbing and Walking Robots*.

Tedrake, R., Zhang, T. W., and Seung, H. S. 2004. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *IEEE International Conference on Intelligent Robots and Systems*.

Teh, Y. W., and Roweis, S. 2003. Automatic alignment of local representations. In *Advances in Neural Information Processing Systems 15*.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 22, 2319–2323.

Thorne, M., Burke, D., and van de Panne, M. 2004. Motion doodles: an interface for sketching character motion. In *SIGGRAPH 2004 Conference Proceedings*, 424–431.

Torkos, N., and van de Panne, M. 1998. Footprint–based quadruped motion synthesis. In *Graphics Interface*, 151–160.

Tyler-Whittle, M. S. 2002. *Gait analysis: an introduction*, third ed. Oxford ; Boston : Butterworth-Heinemann.

Uno, Y., Kawato, M., and Suzuki, R. 1989. Formation and control of optimal trajectory in human multijoint arm movement minimum torquechange model. *Biol Cybern 61*, 89–101.

Unuma, M., Anjyo, K., and Takeuchi, R. 1995. Fourier principles for emotion-

based human figure animation. In *Proceedings of SIGGRAPH 1995*, 91–96.

Vakakis, A. F., and Burdick, J. W. 1990. Chaotic motions in the dynamics of a hopping robot. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 1464–1469.

van de Panne, M., and Fiume, E. 1993. Sensor-actuator networks. In *Proceedings of SIGGRAPH 1993*, 335–342.

van de Panne, M., Kim, R., and Fiume, E. 1994. Virtual wind-up toys for animation. In *Graphics Interface*, 208–215.

van de Panne, M. 1997. From footprints to animation. *Computer Graphics Forum 16*, 4, 211–224.

Vicon. Vicon 6 hardware user manual. Vicon Motion Systems Ltd. `http://www.vicon.com/`.

Vukobratovic, M., and Juricic, D. 1969. Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering 16*, 1–6.

Vukobratovic, M. 1990. *Biped locomotion : dynamics, stability, control, and application*. Springer-Verlag.

Wang, Z., and van de Panne, M. 2006. Walk to here: A voice-driven animation system. In *SIGGRAPH/EG Symposium on Computer Animation*, 243–250.

Wang, T., Dordevic, G. S., and Shadmehr, R. 2001. Learning the dynamics of reaching movements results in the modification of arm impedance and long-latency perturbation responses. *Biological Cybernetics 85*, 6, 437–448.

Winter, D. A., Patla, A. E., Ishac, M., and Gage, W. H. 2003. Motor mechanisms of balance during quiet standing. *Journal of Electromyography and Kinesiology 13*, 1, 49–56.

Winter, D. A. 1995. Human balance and posture control during standing and walking. *Gait & Posture 3*, 4, 193–214.

Winters, J. M., and Crago, P. E., Eds. 2000. *Biomechanics and Neural Control of Posture and Movement*. Springer-Verlag.

Wise, S. P., and Shadmehr, R. 2002. *Motor Control*. Elsevier Science.

Witkin, A., and Kass, M. 1988. Spacetime constraints. In *Proceedings of SIGGRAPH 1988*, 159–168.

Witkin, A., and Popovic, Z. 1995. Motion warping. In *Proceedings of SIGGRAPH 1995*, 105–108.

Wolpert, D. M., and Ghahramani, Z. 2000. Computational principles of movement neuroscience. *Nature Neuroscience 3*, 1212–1217.

Wooten, W. L. 1998. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology.

Wrotek, P., Jenkins, O. C., and McGuire, M. 2006. Dynamo: dynamic, data-driven character control with adjustable balance. In *sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, 61–70.

Xie, M. 2003. *Fundamentals of Robotics: Linking Perception to Action*. World Scientific.

XSensor. X2 model guide. XSensor Technology Corporation. `http://www.xsensor.com/`. 1212–1217.

Yang, C., Sharon, D., and van de Panne, M. 2005. Sketch-based modeling of parameterized objects. In *Second Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 0–0.

Yin, K., and Pai, D. K. 2003. FootSee: an interacitve animation system. In *SCA'03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 329–338.

Yin, K., Cline, M. B., and Pai, D. K. 2003. Motion perturbation based on simple neuromotor control models. In *PG'03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, 445–449.

Yin, K., Pai, D. K., and van de Panne, M. 2005. Data-driven interactive balancing behaviors. In *PG'05: Proceedings of the 13th Pacific Conference on Computer Graphics and Applications*, 118–121.

Yin, K., Loken, K., and van de Panne, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Trans. Graph. 26*, 3, 105.

Yokoi, K., Kanehiro, F., Kaneko, K., Fujiwara, K., Kajita, S., and Hirukawa, H. 2002. Experimental study of biped locomotion of humanoid robot hrp-1s. In *8th International Symposium on Experimental Robotics*.

Zettel, J. L., McIlroy, W. E., and Maki, B. E. 2002. Environmental constraints on foot trajectory reveal the capacity for modulation of anticipatory postural adjustments during rapid triggered stepping reactions. *Experimental Brain Research 146*, 1, 38–47.

Zhao, P., and van de Panne, M. 2005. User interfaces for interactive control of physics-based 3D characters. In *ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*.

Zhao, P. 2004. *Animation Pallette*. Master's thesis, University of British Columbia.

Zhu, C., and Kawamura, A. 2003. Walking principle analysis for biped robot with ZMP concept, friction constraint, and inverted pendulum model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 364–369.

Zordan, V. B., and Hodgins, J. K. 1999. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Graphics Forum (Proceedings of Eurographics 1999)*, 13–22.

Zordan, V. B., and Hodgins, J. K. 2002. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH Symposium on Computer Animation*, 89–96.

Zordan, V. B., Majkowska, A., Chiu, B., and Fast, M. 2005. Dynamic response for motion capture animation. *ACM Trans. Graph. 24*, 3, 697–701.