Real-Time Planning and Control for Simulated Bipedal Locomotion

by

Stelian Coros

B.Comp., University of Guelph, 2006

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

(Vancouver, Canada)

December, 2010

Abstract

Understanding and reproducing the processes that give rise to purposeful human and animal motions has long been of interest in the fields of character animation, robotics and biomechanics. However, despite the grace and agility with which many living creatures effortlessly perform skilled motions, modeling motor control has proven to be a difficult problem. Building on recent advances, this thesis presents several approaches to creating control policies that allow physically-simulated characters to demonstrate skill and purpose as they interact with their virtual environments.

We begin by introducing a synthesis-analysis-synthesis framework that enables physically-simulated characters to navigate environments with significant stepping constraints. First, an offline optimization method is used to compute control solutions for randomly-generated example problems. Second, the example motions and their underlying control patterns are analyzed to build a low-dimensional step-to-step model of the dynamics. Third, the dynamics model is exploited by a planner to solve new instances of the task in real-time.

We then present a method for precomputing robust task-based control policies for physically simulated characters. This allows our characters to complete higher-level locomotion tasks, such as walking in a user specified direction, while interacting with the environment in significant ways. As input, the method assumes an abstract action vocabulary consisting of balance-aware locomotion controllers. A constrained state exploration phase is first used to define a dynamics model as well as a finite volume of character states over which the control policy will be defined. An optimized control policy is then computed using reinforcement learning.

Lastly, we describe a control strategy for walking that generalizes well across gait parameters, motion styles, character proportions, and a variety of skills. The control requires no character-specific or motion-specific tuning, is robust to disturbances, and is simple to compute. The method integrates tracking using proportional-derivative control, foot placement adjustments using an inverted pendulum model and Jacobian transpose control for gravity compensation and fine-level velocity tuning. We demonstrate a variety of walking-related skills such as picking up objects placed at any height, lifting, pulling, pushing and walking with heavy crates, ducking over and stepping under obstacles and climbing stairs.

Preface

Versions of Chapters 3, 4 and 5 have been published in three separate peer-reviewed academic journals. My supervisor, Dr. Michiel van de Panne, played a major role in the writing of the manuscripts. Our numerous discussions were also key to the development of the control frameworks described in this thesis. The relative contributions of my other co-authors are discussed below.

Chapter 3

Published as Coros, S., Beaudoin, P., Yin, K., and van de Panne, M. (2008). Synthesis of constrained walking skills. *ACM Transactions on Graphics (Proceedings Siggraph Asia)*, 27(5). Philippe and KangKang contributed with helpful discussions and prepared the video demonstrating our results. I implemented the control framework, first in 2D and then in 3D. Furthermore, I conducted all experiments, gathered the results and helped with the writing of the academic paper. I also presented this work in Singapore in December 2008.

Chapter 4

Published as Coros, S., Beaudoin, P., and van de Panne, M. (2009). Robust Task-based Control Policies for Physics-based Characters. *ACM Transactions on Graphics (Proceedings Siggraph Asia), 28(5).* I implemented the control framework, worked on a first draft for the paper together with Philippe, gathered most of the results and presented the work in Yokohama, Japan in December 2009. Philippe helped by gathering some results and with the preparation of the final video.

Chapter 5

Published as Coros, S., Beaudoin, P., and van de Panne, M. (2010). Generalized Biped Walking Control. *ACM Transactions on Graphics (Proceedings Siggraph)*, 29(3). I implemented the control framework, helped writing the paper, gathered most results and presented the work in Los Angeles in July 2010. Philippe took part in numerous discussions and implemented a graphical user interface that allows novice users to design humanoid characters of various proportions and create various motion styles.

Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisor, Dr. Michiel van de Panne, for his help and guidance throughout my time as a graduate student. It's been really fun working with you over the past four years. For that, and for everything you have taught me, I am deeply indebted to you!

I want to thank my supervisory committee members, Dr. Dinesh Pai and Dr. Robert Bridson, my university examiners, Dr. Alan Mackworth and Dr. Tania Lam, and my external examiner, Dr. Jehee Lee. Your insightful comments have improved the content of my thesis. Philippe and KangKang, my co-authors, I want to thank you too! Our numerous discussions were key to this work.

Thank you also to all my friends and fellow graduate students. You've created a fun and memorable work environment. A big thanks goes to my lunch and Hapkido buddy, Gordon! It's been a real pleasure.

Finally, and most importantly, I want to thank my wife, Kristen, my parents, Elena and Olimpiu, and my sisters Andreea and Alexandra. I could not have done it without your love and support. Dad, you've always been my biggest inspiration! Kristen, mulţumesc şi te iubesc!

Chapter 1

Introduction

Whistling a tune, Steve is walking home from work after a colossal rainstorm. Children playing soccer on the street catch his attention. With renewed youth, he mindlessly hops over two puddles and kicks a fallen chestnut. "He shoots, he scores!", he mutters to himself. Moments later, distracted by thoughts of dinner, he does not notice a child careening towards him on a tricycle until they almost collide. He leaps onto the grass, out of harm's way, shaking his fist at the retreating cyclist.

Humans and animals are capable of a wide array of skilled motions, all of which are the result of internal and external forces acting on the musculoskeletal system. The internal forces are generated by the nervous system through muscle activations in a process called motor control. The internal forces directly enable us to perform basic actions such as lifting an arm or a leg. Importantly, they also give us the means to indirectly manipulate contact and frictional forces, which allows us to interact with the world in a rich variety of ways. Walking, hopping over puddles, riding bicycles and playing soccer are just a few examples.

1.1 Motivation

There are compelling reasons to create models of the motor control strategies observed in nature, as the applications are far-reaching. Legged robots, which are becoming increasingly prevalent, require robust models of motor control in order to skillfully navigate urban environments that are designed for humans. In fact, everyday environments are just the beginning for such robots, as evidenced by a program launched by the National Aeronautics and Space Administration (NASA) aimed at sending the Robonaut, a humanoid robot, to the Moon [NASA, 2010].

In the fields of biomechanics and physiotherapy, a promising step towards rehabilitation of patients with spinal chord injuries relies on Functional Electrical Stimulation (FES). Using this technology, individual muscles of a paralyzed limb can be activated through electric impulses. Future neuroprosthetic devices that combine FES with suitable motor control models have the potential of being able to restore the ability of paraplegic and quadriplegic patients to stand and walk. Coupled with

realistic simulation of muscles, simulated motor control models could also be used to predict the impact of surgical procedures on a patient's mobility and rehabilitation.

Used in conjunction with physics simulations, motor control models can lead to the creation of realistic computer representations of humans and animals. Once validated on living creatures, such an approach could be used to bring to life extinct or imaginary creatures. Video games and virtual training applications could benefit from autonomous simulated humans that interact realistically with their environments and with each other. For films, modeling motor control strategies could greatly simplify the tedious manual work that animators are currently responsible for, especially for scenes containing hundreds or thousands of characters.

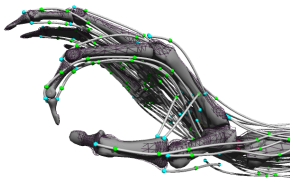
(a) Boston Dynamics' BigDog [Raibert et al., 2008]

(b) Boston Dynamics' LittleDog [BostonDynamics, 2010]

(c) Honda's ASIMO [Sakagami et al., 2002]

(d) Samsung's Mahru [Kwon et al., 2007]

(e) A simulated character being pushed [Yin et al., 2007]



(f) Musculoskeletal control [Shinjiro et al., 2008]

Figure 1.1: Modeling motor control strategies allows robots to interact with the real world and simulated characters to produce realistic, responsive motions.

1.2 State of the Art

The starting point for most frameworks that deal with control of locomotion is basic walking in a straight line. Once this is in place, there are numerous other goals that are sought: energy efficiency; robustness to unplanned disturbances; high motion quality; the ability to get up after a fall, stand, walk and run; controllable motion style and gait parameters such as the walking speed and heading; controller generalization to characters of various dimensions; navigation of rough or slippery terrains; locomotion in constrained environments with obstacles that need to be stepped over, under or

onto; agile completion of locomotion based tasks such as getting to a target location; purposeful interaction with the environment such as picking up, carrying, pushing or pulling objects. Ideally, the control framework should also be easy to implement, fast to evaluate and should work well on real-life robots, or in conjunction with accurate biological models. The grand challenge in the area of control of locomotion is the development of control strategies that can achieve all the goals stated above. In other words, we wish to be able to replicate the capabilities of real-life creatures.

The past few decades have seen considerable time and effort being put into the task of reaching the aforementioned goals, with impressive results being achieved: Boston Dynamics' quadrupedal robots, BigDog and LittleDog, can navigate rough terrains, (Figure 1.1(a, b)); humanoid robots such as Honda's Asimo or Samsung's Mahru III have behavioural repertoires which include walking, running, dancing or climbing stairs (Figure 1.1(c, d)). Inspired by insights into control strategies for robotic systems, researchers in the field of computer animation have presented control methods that allow physically-simulated characters to recover balance in the face of significant unplanned external disturbances while standing, walking or running(Figure 1.1(e)). In biomechanics, much of the emphasis is placed on developing accurate musculoskeletal models that can be used to control movements (Figure 1.1(f)).

Despite these achievements, creating suitable models of motor control has proven to be a vexing problem. The grace, agility and versatility of real-life creatures remain unmatched. There are several factors that contribute to the difficulty of the problem. When viewed from a control theory perspective, humans and animals are high-dimensional, non-linear dynamical systems which makes their mathematical modeling infeasible. These dynamical systems are also under-actuated, as the Center of Mass (COM) can only be controlled indirectly through purposeful manipulation of friction and contact forces. Furthermore, humans and animals are inherently unstable as the points of support, typically the feet, are well below the COM. Even slight disturbances can quickly amplify unless properly corrected in a timely fashion.

The creatures for which we wish to create models of motor control have many muscles, or actuators, that need to be activated in coordinated fashion, continuously through time. The generated control inputs need to take into account signals coming from a variety of sensors. For instance, the vestibular system of the inner ear is used to keep track of the head's motion. Sensory receptors in the skin report information on pressure due to contacts with various objects, and information from the visual system helps create models of the world that allow motions to be planned out. The motions then need to be carefully executed - using too much or too little force in a leap, for instance, could cause a mountain goat to miss the planned landing location, with potentially disastrous results.

1.3 Thesis Overview

The work presented in this thesis explores models of motor control that are suitable for purposeful and agile locomotion skills. We begin with a survey of related work in the fields of character animation and robotics (Chapter 2). Discussions of relevant work also appear throughout the other chapters. Figure 1.2 provides an overview of the structure of this thesis. SIMBICON (Section 2.4), a particularly simple and robust low-level control strategy for basic walking skills, serves as the starting point for our work. The frameworks presented for Constrained Walking (Chapter 3) and Task-Level Control (Chapter 4) use SIMBICON controllers as low-level actions that the planners we introduce reason in terms of. The work on Generalized Biped Walking Control (Chapter 5) then presents an improved low-level control strategy that is loosely based on SIMBICON. Simple scripting rules allow our characters to perform higher-level goals than previously possible with basic locomotion controllers. We conclude the thesis with conclusions and suggestions for future work (Chapter 6).

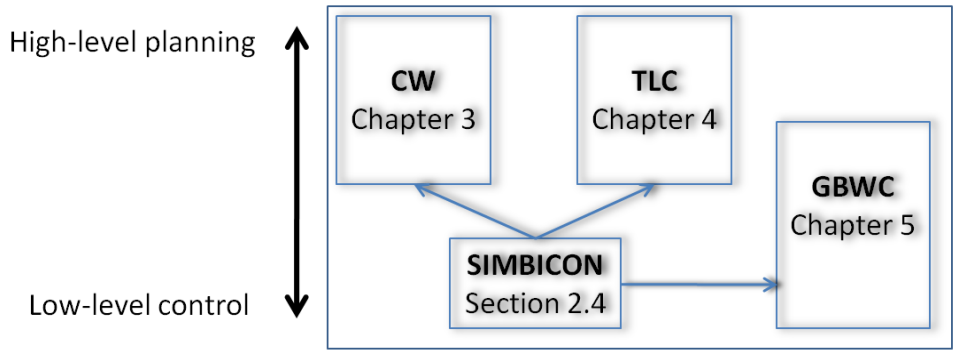


Figure 1.2: Thesis overview. The planners we introduce for Constrained Walking (CW) and Task-Level Control (TLC) use SIMBICON controllers as low-level actions. The Generalized Biped Walking Control (GBWC) strategy, which is loosely related to SIMBICON, allows us to bridge the gap between low-level control and higher-level planning through simple scripts.

1.3.1 Common Elements

In order to meet the goal of modeling purposeful motor control skills, we must consider ways of integrating control strategies with motion planning, as real creatures do all the time. This problem is difficult because an appropriate motion planner needs to be aware of the types of motions a virtual character can perform at any moment in time. In turn, the virtual character needs to know what motor control inputs are needed to enact the motions requested by the planner. In other words, there is a strong level of coupling between planning and control. In addition, an appropriate

planning and control framework needs to work at various levels of detail. There are short-term goals, such as computing the full-body joint torques at each moment in time that will get the character walking at some desired speed, medium-term goals like recovering balance within a couple of steps in case of an unplanned disturbance, and long term goals such as collision-free navigation in a room. The ideas we propose to deal with these problems share a number of important themes.

Abstract Actions. The control strategies we develop use multiple levels of abstraction. *Task-level actions*, represented by low-level, balance-aware locomotion controllers (Section 2.4), are selected by the planner with the goal of generating purposeful, task-driven motions. The low-level controllers chosen by the planner provide *step-level actions*. These actions output *desired joint angles* that are used by Proportional Derivative (PD) controllers to generate *joint torques*. The torques are the output of our control systems, and are used as input into a forward dynamics simulator that is used to generate the motion.

There are significant advantages to having the control policies operate at the level of task-level actions, instead of joint torques directly. The low-level controllers continuously drive coordinated movement, so the motion of each body part does not need to be planned independently. This significantly reduces the dimensionality of the action space our planners have to deal with. The low-level controllers furthermore provide balance-recovering reflexes that are crucial in the face of unplanned interactions with the environment. This enables our planners to be invoked at relatively low frequencies, which significantly speeds up the presented planning and control frameworks.

Task-Relevant Models of the Character Dynamics. We build explicit dynamics models that approximate the dynamics of the characters in relevant regions of the state space. Along with the abstract actions we use, this helps to mitigate the "curse of dimensionality" by not wasting resources modeling the dynamics and control in areas of the state space that are unlikely to be encountered during a task.

Step-Based Predictive Planning Task-level actions are chosen by our planners on a step-to-step basis. The dynamics models we build are used to inform the planner of the actions that are available at any moment in time, and the motions that they are likely to lead to.

With these levels of abstraction in place, two questions remain: How can a suitable action space be obtained, and how can the action space and the dynamics model be used for planning purposes? To answer these questions, we examine two different approaches that are summarized in Sections 1.3.2 and 1.3.3.

1.3.2 Constrained Walking

We first consider the problem of walking reliably in environments with significant foot-stepping constraints, such as gaps that need to be stepped over (Chapter 3). Successful traversals of constrained environments such as the one depicted in Figure 1.3 require skilled walking and planning. Consider, for example, the case of approaching a large gap. The character needs to step close to the near edge of the gap, and, at the same time, it needs to ensure that it has enough forward momentum to take a large step over the gap. The problem is difficult because it is not obvious how it can be parameterized, and because of the wide range of possible scenarios that the control system needs to successfully handle.

Our method takes as input one SIMBICON controller. An offline optimization process automatically modifies various control parameters in order to solve example "stepping stone" problems where the character is told exactly where to step. Through this process, a continuous space of actions that are useful for the task is automatically discovered. The data gathered at this stage is used to create a model of the character's state dependent abilities. At run-time, a planner then uses the abilities model to predict the outcome of sampled actions one or two steps into the future. This allows the planner to select actions that are appropriate given the current state of the character and the configuration of the upcoming terrain.

We first develop the approach on planar bipeds, and then we extend it to a 3d humanoid model. Because the method first produces its own data to work with, we refer to it as a synthesis-analysis-synthesis approach. We speculate that this style of approach also has the potential to be applied more generally to other motion tasks.

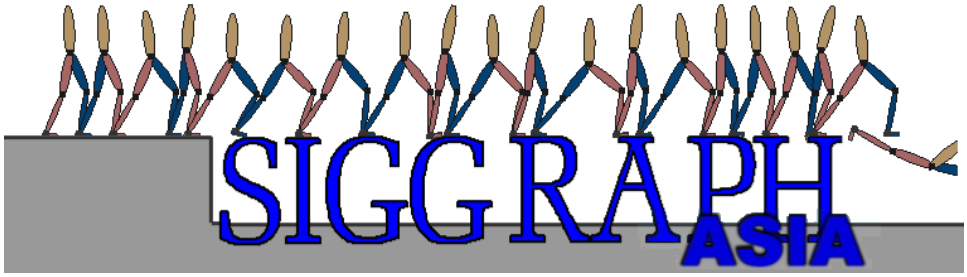


Figure 1.3: Walking in constrained environments.

1.3.3 Task-level Control

A second goal of our work is to develop higher level locomotion skills such as quickly going to a given location or follow a user-directed heading (Chapter 4). This is challenging for a number of reasons. The actions used do not directly specify a motion. Applying a walking controller when a character is running, for instance, will not result in a normal walking step being taken. In addition, significant unplanned interactions with the environment are assumed possible at any time, as shown in Figure 1.4. These interactions can drastically alter the motions that were planned out.

In contrast with the work on Constrained Walking, the input to this control framework consists of a discrete set of low-level controllers which constitutes the action space. We use a state exploration process that sparsely maps the behaviour of each available action in the regions of the state space that are reachable given the input set of controllers. A finite horizon planner could use this data, in a way that is similar to the approach described in Chapter 3. However, in order to get characters that are more agile, we precompute infinite horizon control policies with reinforcement learning algorithms. Real-time results are demonstrates for six locomotion-based tasks and on three highly-varied bipedal characters.

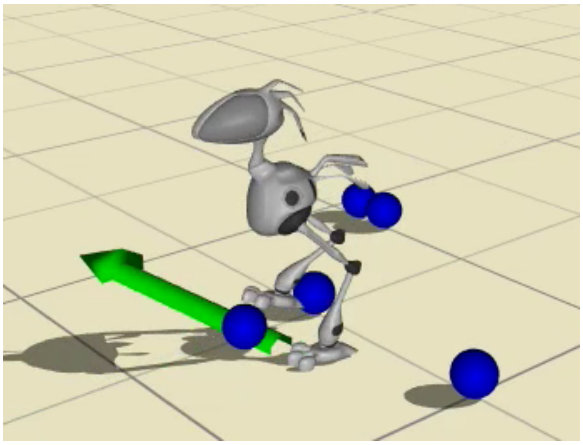


Figure 1.4: Task-level control with environment interaction.

1.3.4 Generalized Biped Walking Control

The last contribution of this thesis consists of a new control strategy suitable for walking-related tasks (Chapter 5). The presented control method requires no character-specific or motion-specific tuning, is robust to disturbances, and is easily parameterized in terms of various high-level gait parameters such as stride frequency, desired speed and step height. We show that our control method generalizes across a variety of walking-related skills, including picking up objects placed at

arbitrary heights, lifting and walking with heavy crates, pushing and pulling crates, stepping over obstacles, ducking under obstacles, and climbing steps. For these reasons, our new control strategy is a more appropriate base locomotion controller than SIMBICON, which was used as a starting point in Chapters 3 and 4.

The method works by integrating tracking using proportional-derivative control, foot placement controlled by an inverted pendulum model and adjustments for gravity and velocity errors using Jacobian transpose control. While the individual components have been known for at least 15 years, their integration is new and provides a surprisingly simple and flexible solution.

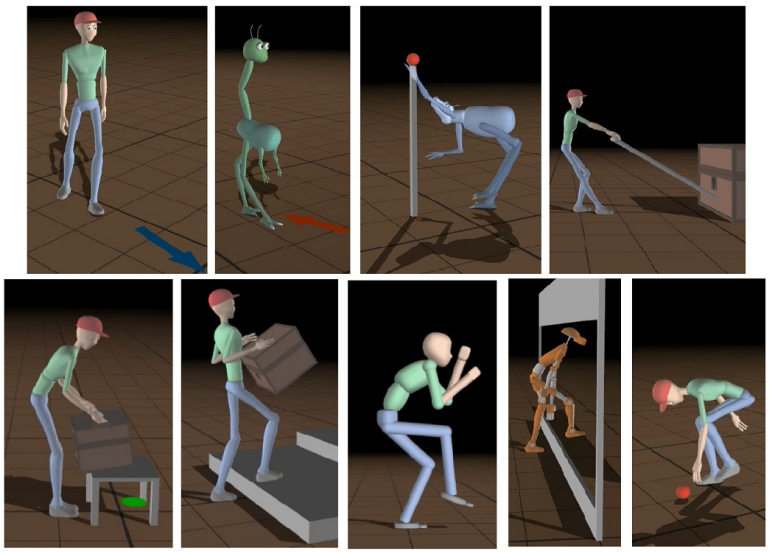


Figure 1.5: Generalized biped walking control examples.

Chapter 2

Related Work

This chapter presents a summary of previous work that is most relevant to our own. We begin with a brief review of relevant work in character animation. We then restrict our focus to relevant control methods introduced in the fields of physics-based character animation and robotics. Lastly, we present SIMBICON, the control strategy introduced by Yin et al. [2007], which is used as a starting point for two of the projects presented in this thesis.

Figure 2.1 shows a bipedal character of the type typically found in video games or other interactive applications. It is represented by a hierarchy of joints and links, with the pelvis being the root of the hierarchy. This character has 33 internal degrees of freedom that parameterize the relative movement of the various joints, and 6 degrees of freedom that describe the character's global position

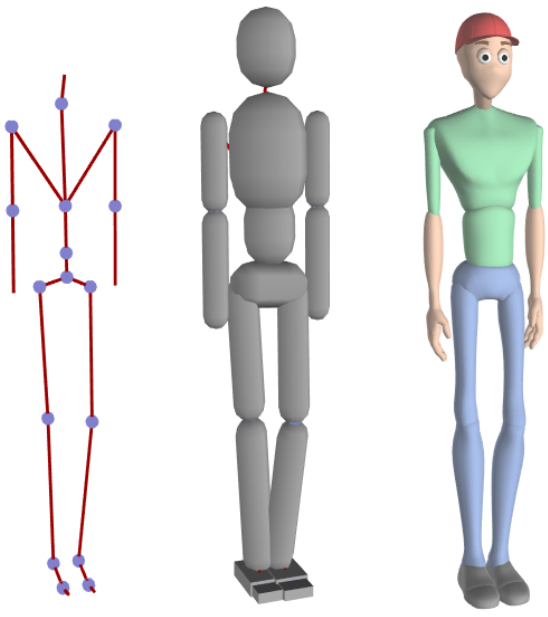


Figure 2.1: A typical bipedal character. Left: joint hierarchy. Middle: collision detection primitives. Right: mesh used for visualization.

and orientation. The motion of a character is represented by time-based trajectories for each of its degrees of freedom. Figure 2.2 illustrates example joint angle trajectories, recorded through motion capture, that define a particular walk cycle.

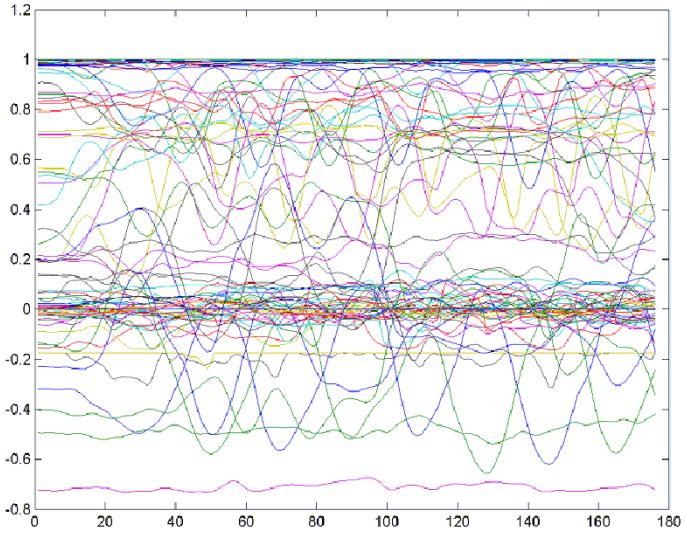


Figure 2.2: Kinematic joint trajectories. Joint angles are plotted as a function of time.

There are several possible approaches to creating animated motions. *Kinematics techniques* re-sequence and interpolate existing joint trajectories that are manually created through keyframing or are obtained from motion capture. *Trajectory optimization methods* automatically create or alter existing trajectories in order to satisfy criteria such as the physical realism of the motion. Lastly, *physics-based animation systems* use models of motor control and forward dynamics simulations to synthesize motions. We now discuss each of these approaches in more detail.

2.1 Kinematic Techniques

Data-driven kinematic models are based on motion data obtained from motion capture or manual keyframing and have become ubiquitous in today's animation systems. Using this straightforward animation model, motions are simply played back when needed. The key insight is that existing pieces of motion can easily be re-sequenced and interpolated in order to create new motions. This idea led to the development of motion graphs, an example of which is illustrated in Figure 2.3. Motion graphs are directed graphs where the edges represent motion clips and the nodes correspond to points where poses from the different motion clips are very similar. Different traversals of a motion graph give rise to smooth, continuous motions that can be significantly different than the original sequences used as input. Motion graphs have been used to animate characters that can,

among other tasks, respond to user input, follow paths through a virtual environment or avoid environmental obstacles [Kovar et al., 2002; Lee et al., 2002; Arikan and Forsyth, 2002; Lau and Kuffner, 2006; McCann and Pollard, 2007; Treuille et al., 2007; Safonova and Hodgins, 2007; Heck and Gleicher, 2007; Beaudoin et al., 2007].

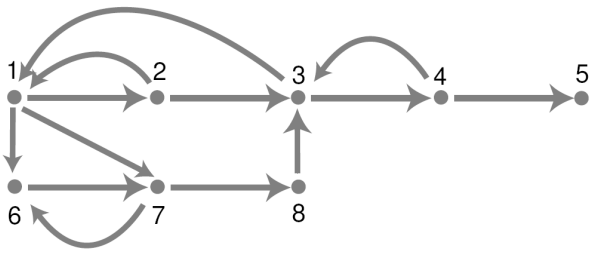


Figure 2.3: A motion graph [Kovar et al., 2002].

Statistical models of motion are based on a deeper analysis of kinematic data. These methods aim at creating probabilistic models of motions that can be used to synthesize new animations[Brand and Hertzmann, 2000; Li et al., 2002], to offer good motion priors and low-dimensional search spaces for trajectory optimization problems [Safonova et al., 2004; Chai and Hodgins, 2007] or to determine the degree of naturalness for new motions [Ren et al., 2005]. Taylor [2008] presents a thorough overview of statistical time-series analysis methods.

Inverse Kinematics (IK) is a commonly-used tool for helping to pose characters for animation. IK automatically computes the joint angles that result, for instance, in the character's hands and feet being placed at specified locations in the world. This allows foot and hand trajectories to be used to help define full body animations. Several animation systems that are based on IK have been proposed: Yamane and Nakamura [2003] introduced a pin-and-drag interface that allows animators to create whole-body motions for humanoid characters. Boulic et al. [1996] presented a method of controlling the center of mass (COM) of a character while it performs other tasks such as reaching for an object. More recently, Hecker et al. [2008] described a system that can be used to animate user-created characters with a wide range of different morphologies. The main difficulty associated with IK based animation is that there is a great deal of redundancy present in the solution space. For this reason, IK is better used in conjunction with data driven animation models [Yamane et al., 2004].

The quality of the motions synthesized through kinematic techniques is ultimately dependent on the amount of data that is used as input. It is, however, impossible to capture the full range of motions humans and animals are capable of, given the virtually infinite ways in which characters can interact with their virtual worlds and with each other. This suggests that kinematic models of motion do not have the ability to create the general purpose, believable characters that are sought

in interactive applications.

2.2 Trajectory Optimization Methods

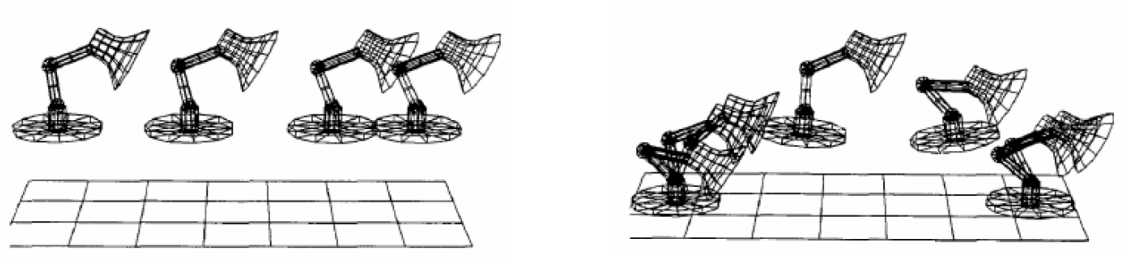


Figure 2.4: Spacetime optimization example [Witkin and Kass, 1988].

Trajectory optimization methods were introduced to the animation community more than two decades ago[Witkin and Kass, 1988], as a new method for creating character animation. The basic premise is that an animator specifies the goals for a motion, and a physically plausible motion is automatically created. In the example shown in Figure 2.4(left), the animator gives the starting and ending poses for Luxo the Lamp, which are treated as constraints for the animation. The laws of physics are also treated as constraints. An optimization method is used to solve for the joint trajectories, and the root position and orientation over time, subject to these constraints. The resulting motion, illustrated in Figure 2.4(right), exhibits desirable features such as follow-through and anticipation that emerge naturally from the optimization.

Trajectory optimization methods have been used to animate simple objects like the Luxo lamp [Witkin and Kass, 1988], hands performing manipulation of objects [Liu, 2009], anthropomorphic characters [Popović and Witkin, 1999; Liu and Popović, 2002; Liu et al., 2005; Jain et al., 2009; Sok et al., 2010], reactions to user-specified external forces [Ye and Liu, 2008, 2010], and creatures with arbitrary morphologies [Wampler and Popović, 2009] among other examples. More recently, Wei and Chai [2010] presented an animation framework that combines trajectory optimization methods with computer vision techniques in order to reconstruct human motions from monocular video sequences.

Trajectory optimization approaches typically lead to very large optimization problems, especially when complex characters with many degrees of freedom are to be animated, or when long motions are desired. In addition, local minima and the choice of optimization criteria used can adversely influence the quality of the synthesized motions, or require some degree of manual parameter tuning. Generally speaking, these methods are also too slow to use in real-time applications and are better suited for off-line animations.

2.3 Physics-based Animation Systems

Another method to generate animated motion is to generate the control inputs - the internal forces or torques - that drive purposeful motions in the real world. Once the control inputs are computed, forward dynamics simulations can be used to obtain the motion of the characters. Advances in this area have the potential of being applicable not only to animating characters in virtual settings, but they may also someday help drive real-world robots.

Despite the ease with which humans and animals have been skilfully controlling their movements for millions of years, modeling appropriate motor control strategies has proven to be a hard problem, especially for bipedal creatures. The character shown in Figure 2.1 has 33 internal degrees of freedom that need to be controlled in order to produce coordinated movement. Since only internal degrees of freedom are directly controllable, the global position and orientation of the characters can only be manipulated through purposeful interactions with the environment. Walking, for instance, is propelled by the friction forces that the feet exert onto the ground. To make matters more interesting, many skilled motions are inherently unstable. As such, a first goal when it comes to control of locomotion skills is being able to maintain balance.

As described by Alexander [2003], balance can be classified as *static* or *dynamic*. During static balance, the projection of the COM on the ground always falls within the support polygon. Small perturbations dissipate with little or no active control, as long the the COM projection does not get too close to the edges of the support polygon. Imagine, for instance, a statue with a flat base. One can push it around to some extent, or place it on a slightly inclined plane without it toppling over.

Quadrupedal animals can locomote while statically stable at all times by using a regular crawl gait. To use this gait, at least three legs should be in contact with the ground at all times, and the COM should always fall within the support polygon defined by the stance feet. However, this gait is rarely used in nature, even by animals such as turtles that have a low center of mass and feet that are wide apart [Alexander, 2003]. Instead, humans and most animals use dynamically stable gaits. Balance is harder to define in this case, since disturbances require active feedback mechanisms. Throughout this thesis we will assume that a character is maintaining balance if it is able to return to a nominal locomotion mode within a few steps. Raibert's hopping robots [Raibert, 1986] and Boston Dynamic's BigDog [Raibert et al., 2008] are examples of robots that dynamically balance themselves.

For the work presented in this thesis we are interested in models of motor control that are capable of reproducing the dynamic balance strategies observed in nature. To date, a variety of such control strategies have been proposed. To evaluate these methods, a wide range of potentially conflicting

attributes could be taken into account, including:

- Robustness to unplanned perturbations
- Naturalness of motion
- Energy efficiency
- Diversity of motions
- Agility and fluidity
- Generalization to different body proportions
- Accuracy of body modeling (ranging from simplified skeletons to accurate musculoskeletal models)
- Robustness with respect to control parameters
- Applicability to legged robots
- Computational performance
- Ease of implementation

It is beyond the scope of this work to precisely categorize previous work according to this list of criteria. Nevertheless, these attributes are useful as we attempt to assess relative strengths and weaknesses for existing control methods below, as well as for the control strategy we introduce in Chapter 5.

Passive dynamic walking

McGeer [1990] observed that a mechanical planar walker could walk indefinitely on slightly sloped terrains without any actuation whatsoever. Stability analysis, however, shows that passive dynamic walkers can only cope with very small perturbations. Successful walks require precisely tuned downhill slopes and initial starting conditions. More recently, several research groups have started investigating the integration of limited actuation capabilities with passive dynamic walkers [Collins et al., 2005; Hobbelen et al., 2008; Pratt and Krupp, 2008].

Policy search and reinforcement learning

A control policy is formally defined as a mapping $\pi : \mathbb{S} \mapsto \mathbb{A}$. In other words, a control policy is used to output an action $a \in \mathbb{A}$, given the state $s \in \mathbb{S}$ of the controlled dynamical system. Control policies can either be directly optimized, or can be computed using Reinforcement Learning (RL) methods that maximize an infinite, discounted sum of rewards [Sutton and Barto, 1998; Ng and Jordan, 2000]. Morimoto et al. [2004] describe an RL-based system that is used to learn appropriate foot placement locations and walk cycle timings for a simplified planar character. Tedrake et al. [2004]

use a stochastic policy gradient algorithm to learn a feedback control policy for a 3D robot with six internal degrees of freedom and four actuators. Byl and Tedrake [2008] compute approximately optimal control policies for a simulated planar robot walking on rough terrain. The simulated robot used in this work has hip actuation and the ability to apply impulsive forces at the toes. The computed control policy selects the desired angle between the hips and an appropriate impulse at the stance toe once per step. Also towards the goal of allowing planar characters to robustly walk on rough terrain, Sharon and van de Panne [2005] present a method that automatically computes non-parametric control policies. The control policies provide state-dependent tracking poses that are obtained through an optimization process. van de Panne and Fiume [1993] and Sims [1994] describe automated methods to search for possible locomotion modes for arbitrary physically-simulated creatures. The resulting motions are appealing, but they do not generally resemble motions of creatures found in nature.

This class of methods is particularly interesting as it investigates automated control policy computation methods. However, the success of these methods largely depends on making appropriate choices in terms of controller architectures, objective functions and parameters to be optimized. Given the many available options, this is not always an easy or intuitive process. To date, these methods have not been shown to scale to control strategies that are robust and versatile enough for the purpose of interactively controlling complex 3D characters.

Simple feedback laws for balanced locomotion

Almost two decades ago, Raibert and Hodgins [1991] introduced some of the earliest work on physically-simulated character locomotion to the graphics community. This work, which was directly influenced by the pioneering work of Raibert [1986], applied control algorithms from robotics to simplified one-legged, bipedal and quadrupedal characters that could hop. Hodgins et al. [1995] extended this work by presenting control strategies for characters with many more controllable degrees of freedom that could perform a variety of behaviours such as running, standing, bicycling and vaulting. Inspired by the insights into hopping and running gaits that were introduced by these projects, Yin et al. [2007] presented a control strategy, nicknamed SIMBICON (SIMple BIped CONtrol). SIMBICON uses a simple, continuous feedback strategy to allow humanoid characters to maintain balance while walking and running, even in the presence of significant unplanned disturbances such as sloped terrain, downward steps and external pushes.

This collection of control strategies uses finite state machines or trajectories derived from keyframes or motion capture data to output target angles for each controllable degree of freedom. The target angles are used to generate joint torques through the use of proportional-derivative (PD) controllers.

Feedback laws are then used to modify the placement of the swing foot, which results in robust locomotion. A detailed description of the control strategy presented by Yin et al. [2007] can be found in Section 2.4.

One disadvantage of these control algorithms is that significant hand-tuning may be required to get a desired motion for a specific character. Various attempts to alleviate this problem have been made. Hodgins and Pollard [1997] presented an automatic algorithm that adapts existing controllers to new characters whose mass, moment of inertia and limb lengths vary. Yin et al. [2007] examined ways of replacing hand-tuned target angles with trajectories obtained from motion capture data, while maintaining the foot-placement strategy. Feedback error learning, as well as an automatic adaptation of the motion capture trajectories, are used to mimic the input motion. Covariance Matrix Adaptation, a stochastic, derivative-free optimization method has been used to automatically tune appropriate controller parameters for characters of various proportions[Wang et al., 2009], or for robustness to uncertain factors such as external pushes, actuator noise and user control inputs[Wang et al., 2010]. Yin et al. [2008] introduced an optimization method for automatically adapting a walk controller to perform very different kinds of tasks such as stepping over or onto an obstacle, walking uphill, walking on ice or pushing heavy objects while walking. This optimization method is particularly successful because it adapts the controllers to a sequence of progressively more difficult tasks.

Control strategies such as the ones mentioned here are well suited for the control of characters that realistically react in response to their environment. Purposeful, autonomous characters, however, need to do more than react. They need to *anticipate* and *plan* their movements in advance. This thesis represents an attempt to integrate such planning behaviours with robust, low-level control strategies.

Motion capture tracking

A popular control paradigm involves the use of motion capture trajectories as target motion that simulated characters attempt to track. To this end, Wrotek et al. [2006] make use of external forces to stabilize the character as it is tracking the input motion. Recently, Liu et al. [2010] present a system that is used to reconstruct the control inputs required to have a physically simulated character follow a stream of motion capture data. The method employs a random sampling based search in order to deal with the lack of derivatives that are hard to estimate in contact-rich motions such as rolls. The control input obtained this way, however, is open-loop. In the presence of external perturbations, the computed solution is likely to lead to failures, so a new solution needs to be found.

To maintain balance while walking, Lee et al. [2010] use inverse dynamics to compute torques that result in a character tracking a motion capture stream. The motion tracked is continuously modified using SIMBICON style linear feedback on the swing hip and stance angle in order to alter foot placement, which leads to increased robustness. da Silva et al. [2008b] also make use of SIMBICON-style feedback, coupled with a short-horizon optimization process that aims to minimize the accumulation of errors in the tracked motion over short periods of time. Sok et al. [2007] allow a planar character to track different motions that can either come directly from motion capture or can be automatically synthesized using an optimization method. Tsai et al. [2009] propose a system where the motion of the upper body is directly tracked, while an inverted pendulum model [Alexander, 1995; Kuo, 1999; Srinivasan and Ruina, 2006] is used in order to determine where the character should be stepping to maintain balance. Kwon and Hodgins [2010] make use of an inverted pendulum on a cart model and captured human motion to simulate running at various speeds.

Trajectory tracking can be viewed as a classical control theory problem, where the objective is to have the character deviate from the input motion trajectory as little as possible. da Silva et al. [2008a] and Muico et al. [2009] use linear and non-linear quadratic regulators to determine time-dependent feedback torques that are meant to optimally eliminate dissimilarities from the tracked motion. Motion planning algorithms designed for data-driven methods, such as the one presented by Treuille et al. [2007], can be used to output motion clips that can be subsequently tracked with such systems. Muico et al. [2009] present a demonstration of a physically-simulated character that walks in continuously changing, user-specified directions. The motions produced with these methods are quite faithful to the input motion capture clips. Robustness to significant external perturbations, however, has not yet been demonstrated, partly because the many linearizations needed by the optimal control methods are only appropriate for small deviations from the input trajectories.

The motions tracked by this class of methods are specific to the person that executed them. It is therefore unclear if these methods can be used to control imaginary creatures or real-life robots. It should also be noted that kinematic motion data only provides joint angles, and does not shed any light on the processes that gave rise to the motion and the available balance recovery strategies. Potentially useful information such as the stiffness of the muscles used to carry out a motion cannot be inferred from the motion alone.

Tracking higher-level motion features

Bipedal motion can, to a large extent, be summarized by a small number of higher level features, such as the location of the center of mass, linear and angular momenta and the position of end

effectors. This has recently been exploited in work that is concurrent to ours. Wu and Popovic [2010] present a control system that uses quadratic programming at every time step to compute full body joint torques that track center of mass and end effector trajectories that are output by a path planner. The path planner's parameters are automatically trained in an offline stage, and take into account user input, variations in the terrain and errors in the desired state of the character.

de Lasa et al. [2010] introduce a prioritized optimization scheme to compute appropriate joint torques at every time step to control characters of various proportions as they stand in place, walk and jump. This work was extended by Mordatch et al. [2010] to include motion planning for rough terrains. A simplified inverted pendulum model is used to approximate and predict various possible outcomes of the character's state at some point in the future. Once an appropriate plan for the inverted pendulum model is found through a stochastic search, the prioritized optimization scheme is used to compute full body joint torques that match the low-dimensional plan.

Motion capture tracking has also been used in conjunction with higher-level motion features. Abe et al. [2007] and Macchietto et al. [2009] treat the problem of tracking motions while maintaining static balance as a multi-objective optimization problem. Constraints on the center of mass, center of pressure and linear and angular momenta can counteract the objective of tracking the input motion, which results in increased robustness. Wu and Zordan [2010] extend the control framework of [Macchietto et al., 2009] to allow for goal-directed or reactive stepping.

Beyond simple locomotion skills

In order to move beyond walking, towards more integrated skills and agile motions autonomous characters need to be able to intelligently string together different actions in order to convincingly simulate the behaviour of humans and animals. To this end, several approaches attempt to combine physical simulation with kinematic-based animation. Zordan et al. [2005] present a system where the motion of characters falling is obtained through simulation, but kinematic techniques are used both before and after the fall. While the simulation is running, the character is actively controlled in an attempt to best match the starting pose of the motion clip that represents a getting up motion. Shiratori et al. [2009] also introduce a system where the character's motion is initially driven by motion capture, but once the character trips, the resulting balance recovery motion is generated through physical simulation. Zordan and Hodgins [2002] and Yin et al. [2003] combine lower-body kinematic motions with upper-body motions that are actively controlled. These systems enable the animation of boxing characters whose motions indicate the impact of incoming punches, or football players that realistically react to balls being thrown at them while performing agile motions such as running. Physically-simulated characters, driven by NaturalMotion's proprietary control

system, Euphoria, [NaturalMotion, 2010] have already started to appear as part of mini games or in specialized scenarios in AAA games such as Grand Theft Auto IV and Star Wars: Force Unleashed. The control strategies used to drive these characters are not made public (they are, however, sufficiently complex that dedicated behaviour engineers need to be deployed with their software).

More relevant to the methods proposed in this thesis are approaches that use solely physically-simulated characters. Wooten [1998] designs controllers that have large enough basins of attraction that direct transitions between controllers is more likely to work. Activities such as a back handspring involve a leap, tumble, land and balance controller to be activated sequentially. Faloutsos et al. [2001] use a series of specialized, hand-designed controllers, that can be combined to create more complex behaviours. The initial conditions for which each specialized controller is expected to work well are learned using support vector machine-based binary classification, and a high-level supervisory controller is tasked with selecting which controller to activate. Relatively complex motions, such as a character sitting on a chair and then standing up, can be simulated with this framework.

2.4 SIMBICON Control Strategy

SIMBICON (SIMple BIped CONtrol) is a control strategy for bipedal locomotion that was recently introduced to the graphics community by Yin et al. [2007] (it is worth noting that this control strategy bears similarity to the work of Raibert and Hodgins [1991] and van de Panne et al. [1994]). SIMBICON has been shown to be remarkably robust to unplanned interactions with the environment, but it requires significant parameter tuning and it provides no means of explicitly controlling higher-level objectives such as the desired speed or the location of a foot placement. The work introduced in this thesis uses SIMBICON as a starting point and presents three methods that are aimed at overcoming some of these shortfalls and enable higher-level control of locomotion-based tasks. Before introducing these methods, however, we describe in this section the details of our SIMBICON implementation.

SIMBICON controller architecture

SIMBICON is at its core a trajectory tracking method where the trajectories can either be user-specified or can be derived from motion capture data as described by Yin et al. [2007]. In our implementation a finite state machine (FSM), or Trajectory Generator, continuously outputs feed-forward target orientations for every joint. The FSM we employ has two states that control the

motion of the character as it swings the left and right leg respectively. Each state has an associated dwell time t, which controls the rate at which the joint trajectories are played out and therefore the time-scale of the motions. Transitions between the two states happen when the swing foot touches the ground, or if the current state has been active for more than t seconds.

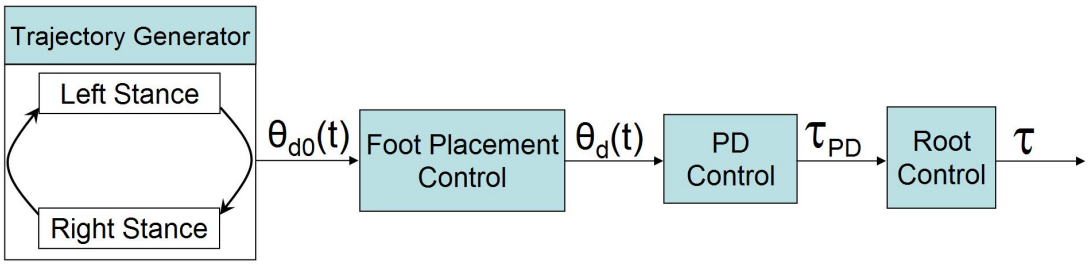


Figure 2.5: SIMBICON system overview. A Trajectory Generator outputs desired joint trajectories, which are modified to enable foot placement control. Proportional-Derivative (PD) control is then used to compute desired joint torques. The resulting joint torques are further processed in order to provide active control to the root (pelvis for 3D characters, torso for planar character).

When it comes to locomotion, one of the most important considerations is being able to maintain balance. To this end, SIMBICON uses two distinct strategies.

Foot placement control. The joint trajectories output by the Trajectory Generator are modified by a linear feedback law that is applied independently in the sagittal and coronal planes:

$$\theta_d(t) = \theta_{d0}(t) + c_d * d + c_v * v \tag{2.1}$$

where $\theta_{d0}(t)$ is the target angle output by the Trajectory Generator, d and v (Figure 2.6(a)) are the distance between the center of mass and stance foot and the velocity of the center of mass respectively, and c_d and c_v are feedback gains that are manually set. Applied to the swing hip, this feedback law has the effect of changing the placement of the swing foot, and has been shown to result in very robust locomotion gaits. The SIMBICON controllers used in this thesis only apply feedback to the swing hip (i.e. c_d and c_v are zero for all other joints).

The joint targets output by the Trajectory Generator specify the relative desired orientation of the child link relative to the parent link. The swing hip desired orientation, however, is specified in world coordinates. This acts as another level of feedback and ensures that, even if the pelvis or torso is leaning forward for instance, the swing hip still raises high enough to ensure swing foot clearance.

Root control. Once obtained, the final sagittal and coronal-plane target angles for each joint are combined into a desired relative joint orientation q_d, which is represented by a quaternion. PD

control is then used to compute appropriate torques for every joint:

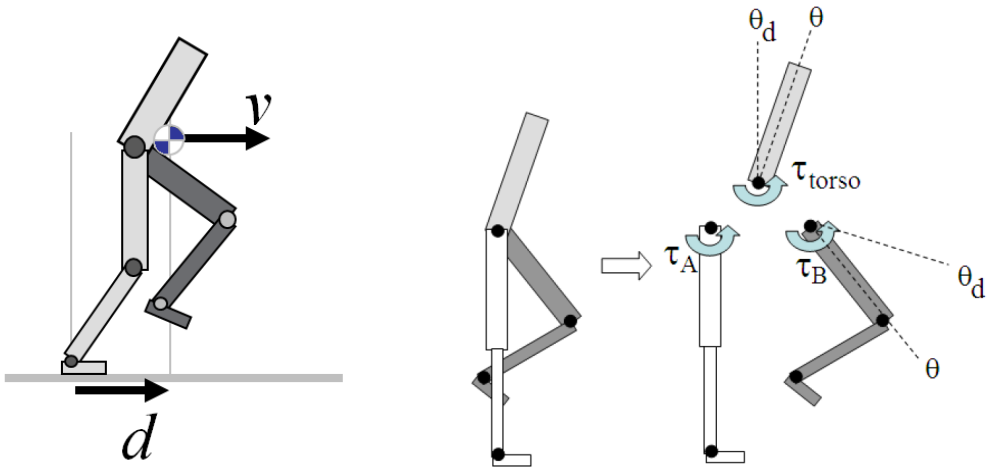
$$\tau_{PD} = k_p * \text{vq}(q_d, q) - k_d * \omega, \tag{2.2}$$

where q and ω are the current relative orientation and angular velocity for the joint, and $\text{vq}(q_a, q_b)$ returns a vector parallel to the rotation axis of $q_a^{-1} q_b$, and equal in magnitude to its rotation angle. k_p and k_d characterize the joint's stiffness and damping properties. The torque applied at each joint is meant to control the orientation of the child link with respect to its parent. The root of the articulated figure representing the character, the pelvis or trunk in our implementation, cannot be controlled in this manner as it is not connected to any parent link. However, it is important that the root's orientation be controlled in order to prevent the character's upper body from leaning over undesirably.

In SIMBICON, the root orientation is controlled through the stance hip. The desired torque, τ_{root}, that the root needs in order to achieve its desired, user-specified orientation is first computed using PD control. A torque τ_A is then applied at the stance hip:

$$\tau_A = -\tau_{root} - \tau_B \tag{2.3}$$

where τ_B is the torque applied at the swing hip, as shown in Figure 2.6(b). With this mechanism in place we lose direct control over the stance hip but ensure a net torque of τ_{root} being applied to the root of the character.



(a) (b)

Figure 2.6: (a) The feedback law used is a linear function of d, the distance from the COM to the stance foot, and v, the velocity of the center of mass. (b) Relationship between torso, stance-hip and swing-hip torques.

Implementation details

The trajectories output by the Trajectory Generator are manually edited in a keyframing like process with the simple graphical user interface illustrated in Figure 2.7. The desired joint trajectories in the sagittal, coronal and transverse planes, specified by Catmull-Rom splines, are edited separately. This simplifies the controller creation process, as it allows the user to focus on editing the motion in the sagittal and coronal planes more or less independently. The feedback law in Equation 2.1 is also applied separately in the sagittal and coronal planes.

The number of control points used to define the Catmull-Rom splines representing the target angles is user defined. In general, we use an average of two control points to specify the trajectory for each internal degree of freedom, and we found we never needed more than six. The PD gains, k_d and k_p are specified per joint, and the balance feedback gains c_d and c_v are non-zero only for the swing hip. The locomotion controllers we designed for the 3D characters we used in this thesis have between 60 and 130 free parameters. For left-right symmetric locomotion gates, only about half as many parameters are required. In general, when designing multiple controllers for the same characters, the PD and balance feedback gains remain unchanged. It is also worth mentioning that many of the control parameters used, such as those specifying the motion of the upper body and arms, are specified mostly for cosmetic reasons.

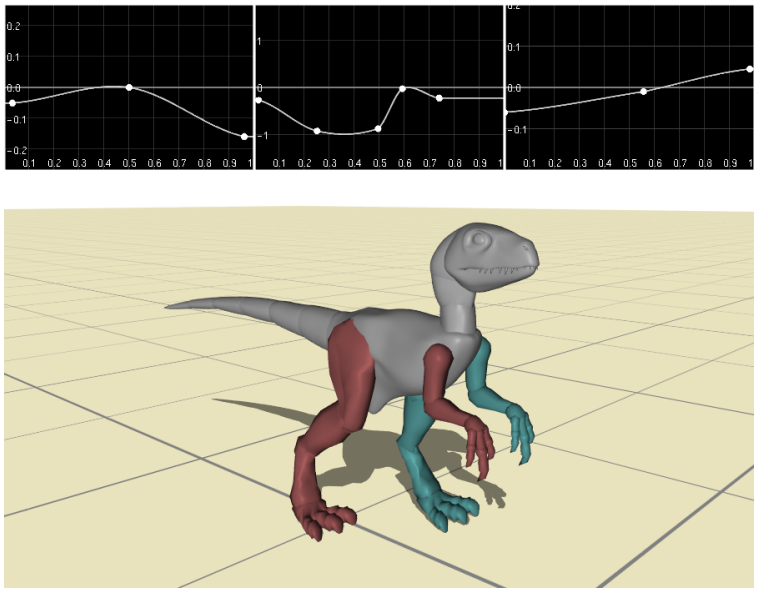


Figure 2.7: Graphical user interface used to author low-level locomotion controllers.

Chapter 3

Constrained Walking

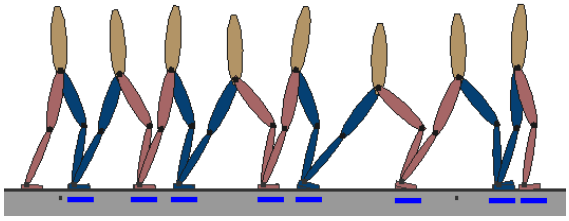
This chapter presents a control framework that allows simulated characters to navigate environments with significant foot-stepping constraints. We begin by automatically constructing a "database" of example scenarios and the actions that are used to solve them. Through this process, a continuous action space that is suitable for the task is automatically discovered. Finite horizon planners then use this information to solve new problems in real-time.

3.1 Introduction

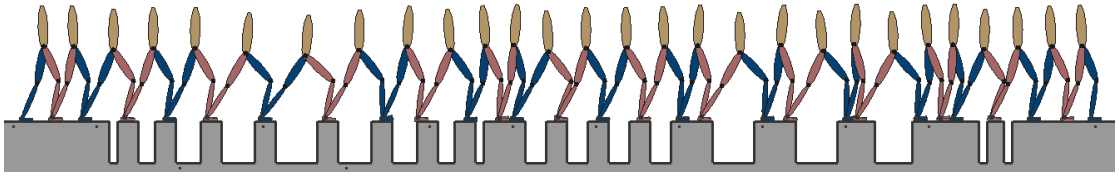
Simulation of skilled walking needs to address issues of balance and control in high-dimensional action spaces, some aspects of which are addressed by previous work. However, skilled walking further requires planning in order to cope with constraints in the environment, such as gaps that need to be stepped over. In this chapter we propose automated techniques for developing constrained walking skills for bipedal characters. As shown in Figure 3.1(b,c,d), the problem is defined as one of walking across a level terrain with gaps, such that the gaps partially or fully constrain where the character can step. A *stepping stone problem* represents a fully-constrained scenario where the sequence of desired foot-step locations has been fixed in advance. The problem is particularly interesting in that there is no obvious parameterization for this task. Modeling the control required as a function of the length of the next desired step is insufficient because the action also needs to take the starting state into account. For example, when taking a $90cm$ step, the character needs a different control input for an initial forward velocity $v = 0.3\ m/s$ as compared to $v = 1\ m/s$. The important role of the character state makes planning and control strongly coupled problems for this task. A planner for walking across a terrain with gaps needs to know what the controller is capable of in any given situation, e.g., in a particular state is it possible to take a large step over an upcoming gap?

Our method is characterized by its *synthesis-analysis-synthesis* approach. First, *offline synthesis* (§3.3) is used to adapt a default physically simulated walking gait to a set of sample problems from

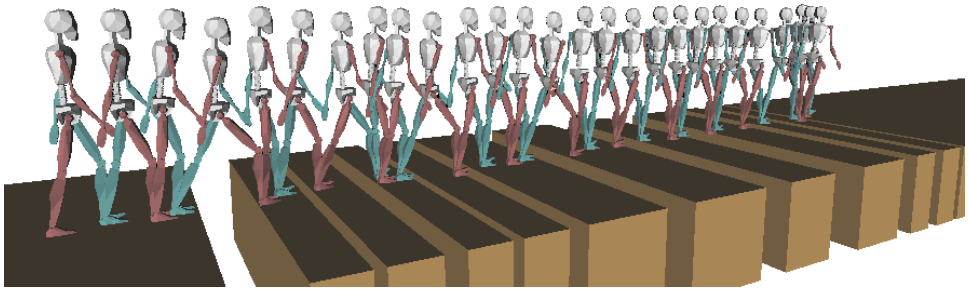
the given task domain. The generated solutions consist of the motions, derived from a forward-dynamics simulation, and the control inputs that created the motions. Figure 3.1(a) shows an example problem sequence of target stepping locations and the simulated motion resulting from the offline synthesis. Second, *offline analysis* (§3.4) is used to develop a low-dimensional step-to-step dynamical model that is based on the family of motions computed in the first step. Third, this dynamical model is exploited during *online synthesis* (§3.5) to plan and control new task instances from the same task domain.



(a)



(b)



(c)

Figure 3.1: Constrained walking skills. (a) Offline synthesis is used to generate physically-simulated motions for example problems. The example motions are used to develop a dynamics model that can make accurate step-to-step predictions. (b) This model can then be used by an online planner to navigate across constrained terrain. (c) Constrained walking for a 3D character.

3.2 Related Work

A variety of previous work touches on aspects of our problem and serve as inspiration for the proposed method. Because of the ubiquitous nature of walking, many data-driven kinematic models of walking have been developed to generate flexibly-parameterized walking gaits [Kwon and Shin, 2005; Mukai and Kuriyama, 2005; Wang et al., 2005]. Several methods demonstrate forms of constrained walking based on resequencing or interpolation of kinematic motion data [Choi et al., 2003; Mukai and Kuriyama, 2005; Reitsma and Pollard, 2007; Safonova and Hodgins, 2007].

Physics-based models are not restricted to using a fixed set of recorded motions thus have the potential to be more general. A variety of feedback-based control strategies have been developed for walking and running characters [Raibert and Hodgins, 1991; Laszlo et al., 1996; Hodgins and Pollard, 1997; Sharon and van de Panne, 2005; Sok et al., 2007; Yin et al., 2007; da Silva et al., 2008b,a]. However, it is not obvious how they can be adapted to handle constrained locomotion scenarios, especially since foot placement is often a key element for regulating balance and speed.

The use of trajectory-based optimization techniques to compute optimized motion sequences has a long history in animation, as exemplified by the work of [Witkin and Kass, 1988; van de Panne, 1997; Popović and Witkin, 1999; Liu and Popović, 2002; Safonova et al., 2004; Liu et al., 2005] and many others. Our offline optimization step is similar to what can be accomplished with trajectory optimization methods in that we desire to modify a default walking gait subject to new stepping constraints. Our proposed method uses the results of the offline synthesis as a point of departure. By observing that there is significant structure in the example solutions, results can be cheaply estimated using regression instead of using an expensive optimization. Unlike the offline optimization problem, solving partly-constrained walking problems requires exploring discrete alternatives in footstep placement, such as whether or not to take an extra step before crossing a gap. As a result, this type of optimization falls outside the scope of gradient-based optimization techniques, even if the stepping location and stepping time are also treated as free parameters. Two last points of distinction of the proposed method are that it can provide real-time control and that it works with standard black-box forward dynamics simulators for both the offline optimization and the online simulation.

Planning physics-based motions across terrain with constraints is a problem that has been studied in the context of hopping robots [Hodgins and Raibert, 1991; Zeglin and Brown, 1998], where it can be simplified in ways that are specific to the structure of this kind of robot. A discrete search strategy is applied to a dynamically simulated hopping lamp character in [Huang and van de Panne, 1996]. Recent work has shown very promising results for terrain-specific policies for planar compass gaits, including terrains with sequences of gaps [Byl and Tedrake, 2008]. Our work is also motivated

by footstep planning for humanoid robotics. Kinematic models of robot capabilities have been used very effectively to compute optimized footfall sequences that avoid obstacles [Kuffner et al., 2003; Chestnutt and Kuffner, 2004]. Kinematic capability models assume that a fixed range of stepping lengths is always achievable. They do not model the fact that a successful large step may be impossible for initial states that are moving too slowly or that a small step may be impossible or undesirable when moving too quickly.

More recent work has demonstrated footstep planning for the Honda ASIMO [Sakagami et al., 2002] robot that does consider the state-dependent nature of actions [Chestnutt et al., 2005]. The planning strategy treats the robot's pre-existing balance control and stepping strategies as a black box that responds to body displacement commands. The key insight is that for the given robot and its controller, the current state of the robot can be accurately predicted by knowing only the last two commanded actions. The planner considers a fixed set of seven possible discrete actions for each step. A sequence of all $7 \times 7 \times 7 = 343$ command sequences is then used to create a discrete model of the space of all possible motions using a directed graph. Motion planning then uses A* search.

Also closely related is the work of [Hofmann, 2006], which demonstrates a motion planning algorithm for a 3D humanoid model for a task involving a prescribed irregular foot placement. The motion is controlled using a user-developed *qualitative state plan* which is a finite state machine abstraction with specified constraints and goals associated with each state. The center of mass position and velocity are used as the state of an abstracted virtual model, which allows for some flexibility when executing the plan.

3.3 Offline Synthesis

Our approach begins by computing dynamically-simulated solutions to example stepping-stone problems, as shown in Figure 3.2. Given a sequence of target foot locations, the goal is to find a control sequence that results in the character stepping precisely at the desired locations. Both the synthesis and the subsequent analysis use individual steps as the basic motion primitive. Steps are defined based on foot-strike events, i.e., one step ends and the next one begins when the swing foot strikes the ground. The output of the process is a sequence of example steps where, for each step i, we know the starting state s_i, the applied action A_i, the resulting state s'_i, and the resulting step length, l_i. The sequence of actions are the unknowns for the example problem sequences. The step length is measured as the heel-to-heel distance.

Finding solutions to a given stepping-stone problem is cast as an optimization problem. Given a base controller defined by parameters A_{base} that produces a steady-state walking gait, the goal of the

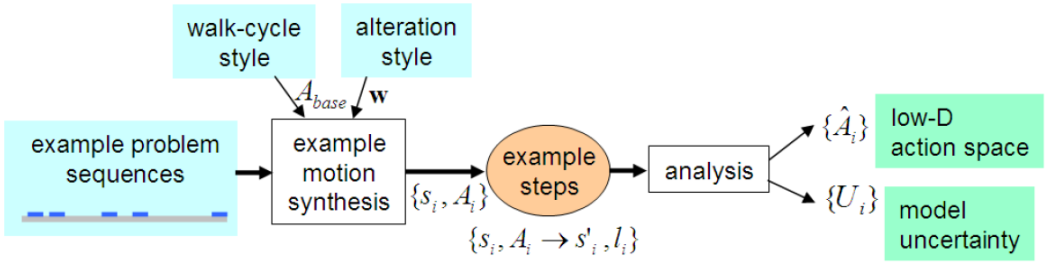


Figure 3.2: Offline synthesis and analysis.

optimization is to find modified parameters for each step, A_i, such that the target stepping sequence is achieved. Implementing these ideas requires defining the example problems, the parameterized base controller, the optimization function, and the optimization method. We now discuss each of these aspects in turn, focusing on their application to planar models. The specific extensions to 3D control are deferred to §3.3.3.

Example stepping-stone problems are generated using a uniform step-length distribution $l \in [0.1m, 0.9m]$ for the humanoid biped and $l \in [0.1m, 1.0m]$ for the big-bird character to be introduced later. We use multiple 100-step sequences instead of a single long sequence because it is possible for the optimization to fail to find good solutions for difficult or perhaps impossible sequences of steps. While such failure is rare, it can compromise solutions for the remainder of the steps in a sequence. The character geometry, control representation, joint limits, and torque limits will all impose constraints on the types of step sequences that are feasible, as will the optimization technique itself.

3.3.1 Actions

The applied actions, A_i, for our method are defined in terms of the finite-state machine (FSM) control structure described in Section 2.4. The FSM consists of two states, one for left-stance and one for right-stance. Transitions between these two states happen once the swing foot strikes the ground. The time-scale of the motion represented by each state is controlled by the state dwell time T_{hold}. Each state uses Catmull-Rom splines to output time-dependent target angles for each joint. The splines representing the target angles for the trunk and swing hip have two control points. The stance ankle target angle is constant and is therefore modeled using only one control point.

Our action vector, A, consists of a set of six parameters of the above controller that can be modified as needed for each walking step. These parameters are the control points for the splines representing

the desired angles of the trunk, swing hip and stance ankle, as well as the FSM state dwell time T_{hold}. The controls for each step are initialized to $A_i \leftarrow A_{base}$, where A_{base} produces a regular walking gait with $36cm$ steps for the humanoid biped. The big-bird character uses the stance knee angle in the optimization instead of the stance ankle angle, and takes $66cm$ steps.

3.3.2 Optimization

Given an example stepping stone problem, the cost function to be minimized by the offline optimization assigns a cost to both stepping-location errors and deviations from the original control parameters:

$$f(A_1 ... A_n) = \sum_{i=1}^{n} (||x_i - x_i^d||^2 + \gamma (A_i - A_{base})^T \mathbf{W} (A_i - A_{base}))$$

where n is the number of steps, x_i^d is the desired stepping location for step i, x_i is the actual stepping location for step i, $\mathbf{W}$ is a diagonal weighting matrix, and $\gamma = 0.1$. We currently use $\mathbf{W} = diag(1, 1, 1, 1, 0.05, 4)$, where these weight the 6 control parameters in the order described above. Distances are measured in metres, angles in radians, and time in seconds. A multitude of optimization methods can now be considered. Simultaneous optimization of the many steps in a long sequence of steps is impractical; a sequence of 100 steps will have 600 free parameters for a problem where analytical gradients are not readily available. Sequentially optimizing one step at a time does not provide sufficient anticipation – the state resulting from a step may be incompatible with being able to achieve the following step.

As a compromise, we optimize over a three-step sliding window that thus has a total of $3 \times 6 = 18$ free parameters. The sliding window is moved forward one step at a time, meaning that any given action A_i for step i will be optimized three times in succession, each time with a different placement in the sliding window. The action for a given step is not finalized until the sliding window has completed all three passes, and only the final resulting action is retained for later use. After a given window optimization is complete, the final parameters for the last two steps in the window become the initial parameters for the first two steps of the next window placement. The state at the end of the first step of the optimization window becomes the new immutable starting state for the next optimization window.

For each position of the sliding window, gradient-descent optimization is used. Although the optimization window spans events that introduce state discontinuities such as foot strikes, the objective function generally varies smoothly as a function of the optimization parameters for our problem domain. We note that unlike the problems tackled in [Yin et al., 2008], our terrain is flat, has constant friction, and is obstacle free. The failure cases that do arise are discarded, as will be

discussed shortly. The gradient is numerically computed using centered finite-differences, and thus makes use of 18×2 simulations spanning the duration of the sliding window. A bisection line search is used to find the local minima in the direction of the gradient, and the gradient-descent operation then repeats. The process stops when the objective function ceases to improve or after 15 iterations have elapsed. The optimization time grows linearly with the number of steps considered in the sliding window.

It is possible to pose stepping stone sequences that are impossible for the optimization to solve to a desired degree of accuracy. This can result in two possible outcomes. The character may end up losing balance and falling, in which case we terminate the stepping sequence and remove the data associated with the five previous steps. Alternatively, the character may end up performing badly in terms of the proposed objective function. We note however that these latter cases still result in valid training data samples. In the data collection phase we are interested in the actual outcome of an action and not necessarily in the desired outcome that was used to generate it. The optimization is nevertheless important because it shapes all the stepping actions in a consistent fashion, thereby giving the action space a considerable amount of structure which we later rely on.

As shown in Figure 3.2, the user has three avenues by which to influence the stepping behavior. The problem stepping sequence specification determines the range of step lengths to be accommodated. The base-control parameters A_{base} determine the walking style. Lastly, it is possible to influence the way in which step adaptations should be made by altering $\mathbf{W}$ or by choosing a different parameterization of the base controller.

3.3.3 3D Control

The offline synthesis process applied to our 3D character is largely identical. We use three control points for the splines that output the target angles for both the the torso and swing hip in the sagittal plane, and two for the stance ankle. These parameters, together with the FSM state dwell time form the action space A.

In the objective function, x and x_{targ} are measured in their projection to the character's sagittal plane, as defined by the root link coordinate frame. We use $\mathbf{W} = \mathbf{I}$. We add one additional term to the objective function that measures lateral step deviation, i.e., $(z_i - z_{targ})^2$, where z_{targ} defines a desired lateral foot spacing of $15cm$. While there are no explicit parameters in A to directly affect lateral stepping, this term discourages the use of subspaces of A which introduce unnecessary lateral disturbances. Single-sided finite differences are used for the 3D case. The 3D offline optimization requires 2.5 minutes per example step. The 3D simulation runs $3\times$ faster than real time. For

comparison, the 2D simulation runs 5× faster than real time.

3.4 Motion Analysis

Motions are planned on a step-by-step basis using an abstract model of the step-to-step dynamics. Given the current state, the planner evaluates the state resulting from each of many possible actions for the current step. This can be applied recursively to look two or more steps into the future. In support of this, the example data is used to build a *step-to-step dynamics model (SSDM)*. As shown in Figure 3.3, the model predicts the state at the start of the next step, $\tilde{s'}$, as a function of the state at the start of the current step, s, and the applied action during the step, A. It also predicts the resulting step length, $\tilde{l}$, and the uncertainty, $\tilde{U}$, of its prediction. The offline data is also used to predict the subspace of reasonable actions that can be taken from a given state s, which we refer to as the *capabilities model*.

The SSDM makes its predictions based upon the example steps computed during the offline synthesis. We employ k-NN interpolation as a simple form of non-parametric regression. Direct application of this in the high-dimensional state space of $s \times A$ yields poor results and ignores the fact that both the states and actions of the example steps exhibit significant structure. To this end, we manually define a lower-dimensional state space, and use PCA to define a lower-dimensional action space. The low-D action space also plays an important role in the planning process by providing a compact action space to sample from, i.e., that defined by $\hat{A}$, as opposed to having to draw samples from the original high-D action space. We now describe in more detail how each aspect of the SSDM is defined.

Low-dimensional state space: The state s is 18-dimensional for planar characters and much higher for the 3D human model. In order to simplify functions that operate on the character state, we define a reduced dimensional representation of the state given by $\hat{s_i} = (d, v, \theta_{torso}, \theta_{Lhip}, \theta_{Rhip})$, where d, v are the position and velocity of the center of mass as measured with respect to the stance foot, and the remaining parameters are the torso, left-hip, and right-hip angles. These features are motivated by the need to model the state in a compact fashion but still capturing the essence of the state. The specific joint angles that we use are therefore those that drive the heaviest links in the character. It should also be feasible to use an automatically-computed low-dimensional state space representation, although we have not yet explored this. We define the distance between two different states s_i and s_j according to $d^2(s_i, s_j) = d^2(\hat{s_i}, \hat{s_j}) = (\hat{s_i} - \hat{s_j})^T \mathbf{X} (\hat{s_i} - \hat{s_j})$ where $\mathbf{X}$ is a diagonal weighting matrix. We use $\mathbf{X} = diag(2, 1, 0.5, 0.5, 0.5)$ for all our characters and styles. For the case of 3D characters, all parameters are identical, but taken in a sagittal projection.

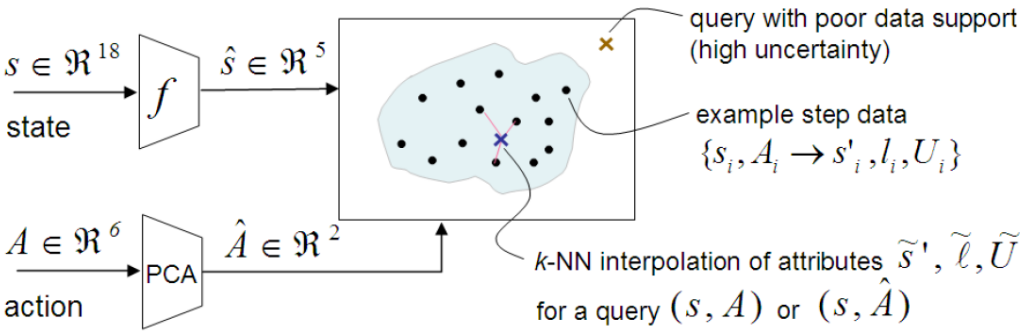


Figure 3.3: The step-to-step dynamics model (SSDM). The non-parametric (example-based) model makes predictions using the results of the offline synthesis. The given dimensions for the state and actions spaces are for the 2D bipeds.

Low-dimensional action space: A principal component analysis (PCA) of all the example actions reveals that there is significant structure in the control actions computed by the offline optimization. 66% of the variation is contained in the first two principal components. We use these first two principal components to define a latent 2D action space $\hat{A}$, whose purpose is to define a unique 2D parameterization for the 6D action space (9D for the 3D character). We project from A down to $\hat{A}$ using the PCA matrices. Where necessary, we estimate A from $\hat{A}$ using kNN interpolation, as will be described shortly. In practice, this gives reconstructions that are better than the 66%-of-variation PCA reconstruction.

kNN regression for SSDM: During planning, the outcomes of many different actions are explored for the current state. In order to efficiently support repeated queries involving the same state s, the regression process first finds the subset of K example steps that have a starting state s_j most similar to s, as measured by $d(s, s_j)$. This subset can be reused for subsequent queries involving the same state. We use $K = 25$. A kD-tree is used to efficiently find the K examples, yielding $3\times$ speedup over straight linear search. The second stage of the regression prediction selects the k nearest neighbors from K based on their distance in reduced action space, $||\hat{A}_i - \hat{A}||$. We use $k = 3$. Lastly, we compute the weights for each of the k samples using $w_i = 1/(d(s_i, s) + \alpha||\hat{A}_i - \hat{A}||)$, followed by a normalization step. The final weights thus take into account distances in both state and action space. We use $\alpha = 1$. Interpolation is carried out using $\tilde{v} = \sum_i w_i v_i$, where v is the value we wish to see interpolated as a function of the query $(s, \hat{A})$. We expect that alternative regression procedures such as Gaussian process latent variable models would likely produce similar results.

During planning, the SSDM is used to predict the resulting state, the step length, and the uncertainty of its own prediction. Once the planner has committed to an action, the SSDM is also used to

estimate the full dimensional action, A, that corresponds to $\hat{A}$. This then becomes the action to be applied. Estimating A using $\hat{A}$ as a latent variable has two advantages over the alternative of using linear reconstruction from the related PCA matrices. It allows A to be modeled as a curved manifold in the high-dimensional space, and it ensures that the model always interpolates and never extrapolates.

Uncertainty model: The uncertainty U provides a way for the SSDM regression to express doubt about the values that it is being asked to estimate. As described in the following section, the planner avoids actions that have uncertain predicted outcomes, either by eliminating them from consideration, or, for fixed-stepping scenarios, adding a penalty cost. For each example step s_i, A_i, s'_i, l_i, we associate an uncertainty estimate U_i, which is computed using leave-one-out cross-validation. We temporarily remove the data for example step i from the set of examples and then use the SSDM to estimate the step resulting from $(s_i, \hat{A}_i)$. This produces $\tilde{s}'_i$ and $\tilde{l}_i$ as estimates of the resulting state and step length, respectively. A comparison of these with their known values is used to compute the uncertainty, which we define as $U_i = d(s_i, \tilde{s}_i) + \beta|l_i - \tilde{l}_i|$. We use $\beta = 1$.

Capabilities model: The example data is also used to provide a model of the subspace of reasonable actions that should be considered when in a given state. This subspace of actions is then used in the planning process. The subset of K example steps having s_j closest to s is used for this, i.e., the same set used in the first stage of the kNN regression. The feasible subspace of actions is defined by the axis-aligned bounding box placed around the K actions in the reduced action space, as illustrated in Figure 3.4. More generally, the convex hull could also be used.

3.5 Motion Planning and Execution

Given a good step-to-step dynamics model, a planning algorithm can use this model to accomplish its task. The goal of the planning can be to return the first satisfactory solution, or, alternatively, to return the best solution according to an optimization criterion. Given the constraint of wanting to control simulated characters in real-time, we opt for either finding the first satisfactory solution, or using the best solution that is found within a fixed number of samples of the action space. We explore three types of planning algorithm, each of which samples from the feasible space of actions as defined by the capabilities model. Each algorithm can plan over a multiple-step horizon, as illustrated in Figure 3.4. Unless otherwise noted, we use a two-step planning horizon. Replanning occurs after each step.

Samples drawn from the space of feasible actions can still result in a number of unacceptable outcomes. The planners reject samples having too much uncertainty, $U \geq U_{max}$ ($U_{max} = 0.75$), or

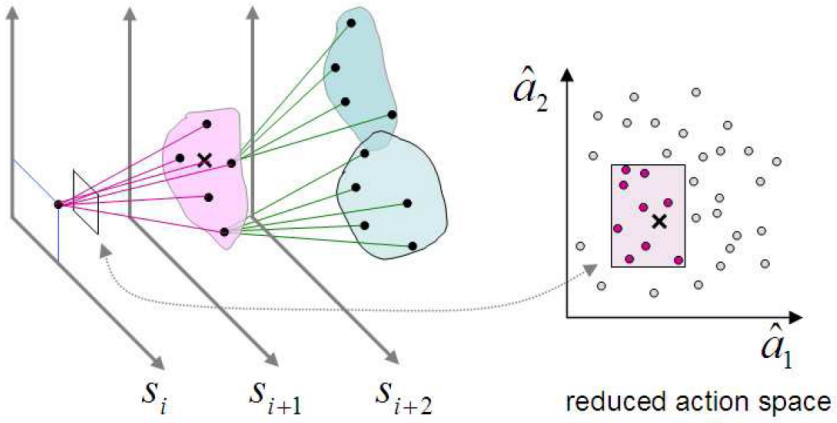


Figure 3.4: Two-step finite horizon planning using the reduced action space. Abstractions of future states considered by the planner are shown.

which have an associated predicted step length which results in stepping into a gap. For stepping-stone scenarios, a sample can be rejected for stepping too far from a desired location. We use a threshold of $6cm$ for this. In order to make a final choice among multiple acceptable options, we use a weighted sum of the predicted step error (as summed over the planning horizon) and the uncertainty associated with the immediate action. We weight the uncertainty with a constant $c_u = 0.1$.

We consider three planning techniques.

Regular Sampling: A first possible planning algorithm is to regularly sample the feasible action space, which can be done recursively until a satisfactory plan is constructed for the next n steps. This technique thoroughly examines the action space and is thus our method of choice for fully-constrained stepping-stone problems, as shown in Figure 3.7.

Random Sampling: For less-constrained problems, such as terrains with sequences of gaps, exhaustive sampling of the action space is typically not required. In such cases we resort to random sampling. For any given step, the random sampling terminates when a satisfactory solution is found or a maximum number of samples for that step has been reached. When applied in the context of building an n-step finite horizon planning, the search operates in a depth-first fashion, and the first solution to successfully achieve n steps is used. The traversal shown in Figure 1.3 is produced using this planning technique.

All planning for the 3D character uses the random sampling approach in order to achieve real-time planning. In order to converge to a regular gait in the absence of obstacles, the base controller A_{base} is used whenever the nearest impending gap is more than $1m$ in front of the character. Otherwise,

random sampling with a two-step horizon is employed, using an upper bound of 500 samples. If this fails to produce a solution that strictly avoids the gaps, the solution that best avoids the gap is chosen.

Hybrid: One aspect missing thus far from our planner is the notion of a preferred step length. In order to incorporate this, we develop a hybrid model. A simple footstep planner is first used to determine the lengths of the next two steps to be taken. Steady-state step lengths are preferred ($36cm$ for the 2D and 3D humanoid bipeds, and $66cm$ for big-bird). If a planned step location falls within a gap, it is moved to either before the gap or after, depending on which edge is closer. Given the footstep plan, a first attempt is made to match the planned steps using a sparse regular sampling in the reduced action space. In the absence of an acceptable solution being found, the footstep planning is abandoned and the random sampling planner is invoked.

3.6 Results

Parameter settings: Our 2D simulation uses optimized Newton-Euler equations of motion and a damped-spring penalty method ground contact model ($k_p = 100000N/m, k_d = 6000Ns/m$) with a time step of $0.0005s$. We compute up to 2000 example steps in order to be able to evaluate the effects of the number of example steps on the quality of our solutions. An average of 11.9 optimization iterations were required per step. For a randomly selected run of 100 steps, the average error in the desired foot placements is $4.3cm$, with 77% of the step errors being less than $6cm$, and 9% more than $12cm$. It is worth noting that not all sequences of steps can be satisfied. The 3D simulation uses the Open Dynamics Engine [ODE, 2010] physics simulator and we generate 1000 example steps. All of the parameters described above for the 2D bipeds are qualitatively similar for the 3D biped. Examples computed using the random-sampling and hybrid planners run in real time, which encompasses most of our examples. Two exceptions are the result shown in Figure 3.1(b) and Figure 3.7, which each use a higher sampling rate during planning in order to find feasible solutions to these highly-constrained problems.

Highly constrained walking: The 2D and 3D simulated characters can plan their way across highly constrained terrains. Figure 3.7 shows the result of a stepping stone traversal, which fully constrains the desired foot locations for each step. Smooth walking involving a mix of small and large steps requires anticipation and this is provided by the planner. The sequence of steps is different from any it has seen in the example data. The error for the last step is $8.9cm$, which is almost twice our average stepping error of $5cm$. The regular-sampling planner is used for this particular example, which does not run in real-time.

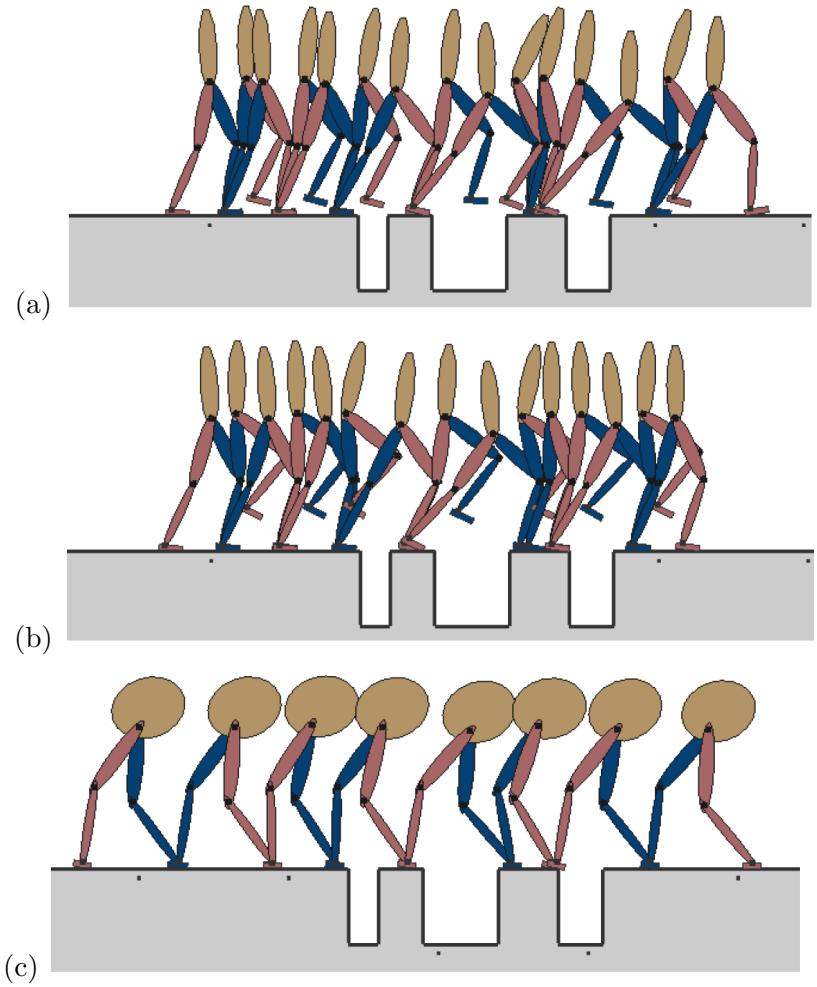


Figure 3.5: Walking Style Examples. (a) Regular walk. (b) High-stepping walk. (c) Big-bird walk.

For the highly-constrained terrains shown in Figures 3.1(b) and (d), the planner must decide where and how to step. The hybrid planner is used for both examples and runs in real-time for the 3D model terrain traversal (Figure 3.1(d)). The largest gaps for this latter example are $50cm$ wide and therefore require at least a $70cm$ step in order to safely cross with a $20cm$ foot. The 2D result shown in Figure 3.1(b) runs slower than real time because of the high sampling rate needed in order to find a feasible solution to this particular problem.

Styles and Characters: Figure 3.5(a) shows a result for our primary model, a 7-link planar biped. A significant feature of our method is that it can be readily applied to alternate walking styles and alternate physical models without making any changes to the synthesis pipeline or any of its parameters. Figure 3.5(b) shows the terrain traversal simulation that results from using a base walking style that lifts the swing leg much higher during mid-stance. This same style is preserved in

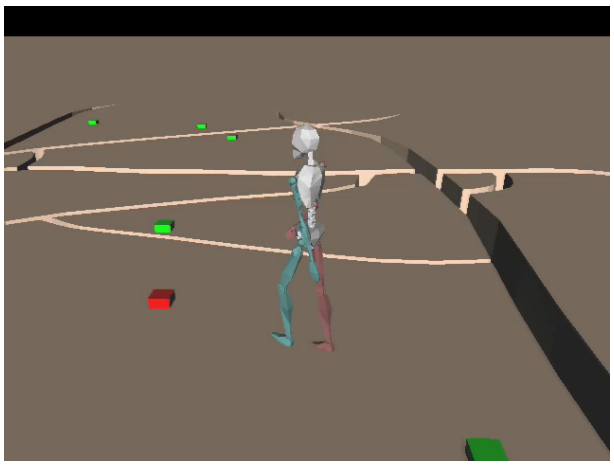


Figure 3.6: Following a path while avoiding crevasses.

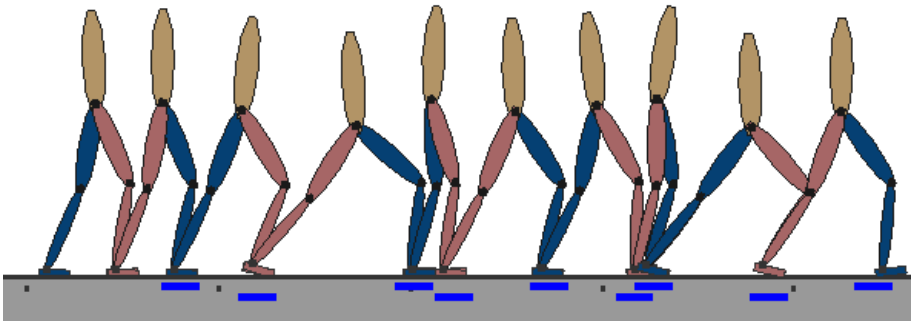


Figure 3.7: Results for a stepping-stone problem.

the resulting simulated motions. Similarly, we can apply the synthesis pipeline to a new character, such as the big-bird character shown in Figure 3.5(c). The same synthesis-analysis-synthesis steps are applied, with no changes to any of the parameter settings. Interestingly, the strategy that emerges to deal with constrained foot-placements for the big-bird character is quite different from that of the human-like biped. Qualitatively, it accomplishes much of the required constrained stepping by having the body move at a relatively constant speed and stepping faster when constraints require taking short steps. This is an effective strategy and we speculate that it may result in part from the small feet of this creature. Figure 1.3 shows a highly constrained walk that can be planned in real time.

3D Path Following: Figure 3.6 is an example of following a path while using real-time planning to step across crevasses. The path is defined using a sequence of way-points. Turning towards the way point is accomplished on any given step using the stance hip, as described in [Yin et al., 2007]. Once within $50cm$ of the current waypoint, the next waypoint becomes the goal.

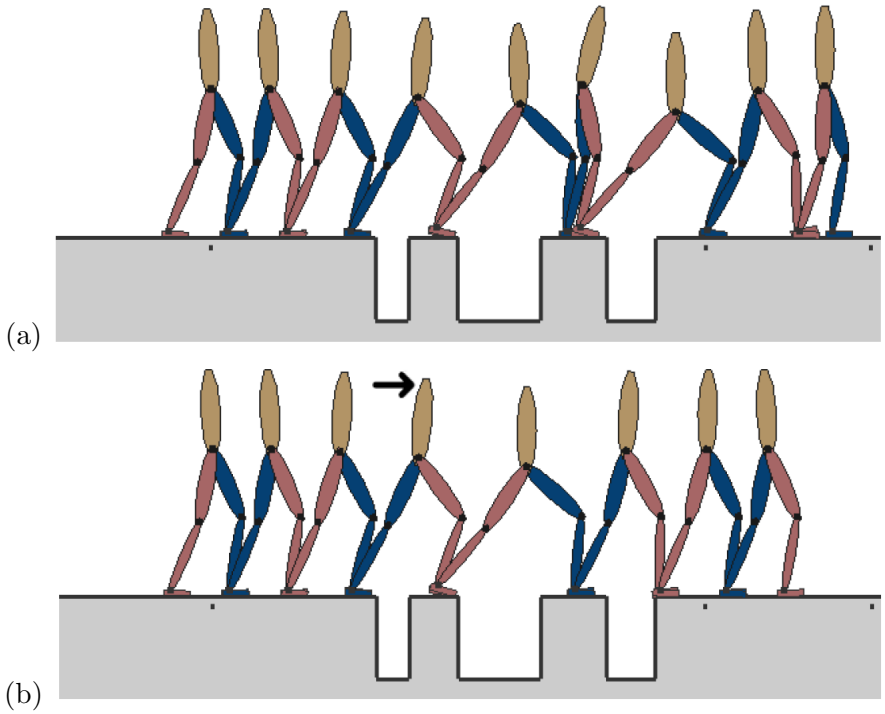


Figure 3.8: The effect of a push on the result of a terrain traversal simulation. (a) Resulting motion with no push. (b) Resulting motion with a push. The extra forward speed from the push results in only a single step being taken on the terrain before the last gap.

Interaction and Replanning: Replanning at every step allows for interactive unplanned physical interaction of the characters (planar and 3D) with their environment. Figure 3.8 shows how a mid-stride push will affect the resulting motion. The use of a continually-active balance mechanism [Yin et al., 2007] adapts the placement of the swing foot without delay, although of course an unfortunately timed push could in this way cause a step into a gap. Upon foot-strike, the planning process takes the current state into account when developing its subsequent plan. We also show robustness to small changes in terrain height ($4cm$) while stepping over gaps, as well as an example of the 3D model adapting to a push. While the pre-existing balance mechanism provides the immediate response, it is the step-by-step motion planning that results in the required adaptation with respect to upcoming gaps.

Effect of number of example steps: The locomotion skill of the character is in part a function of the number and span of the motion prototypes that are computed offline. Figure 3.9 shows the distribution of foot-placement errors for a fully-constrained stepping stone walk across a new stepping-stone sequence for the planar human model. The ability to precisely follow a given stepping-stone sequence improves with more example data. The performance figures are for a terrain that has the same uniform random distribution of requested step lengths as was used for generating the

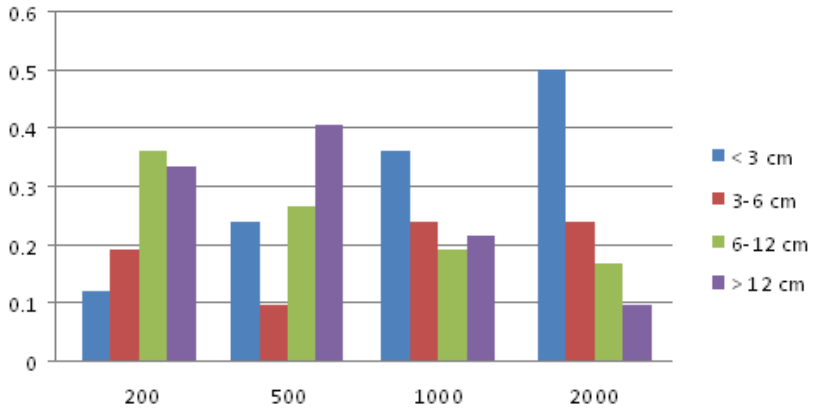


Figure 3.9: Performance as a function of the number of example steps. The colored histograms give the distributions of stepping length errors when using the given number of synthesized example steps.

example data.

Effect of planning horizon: The effect of the planning horizon is shown in Figure 3.10 as evaluated on the planar human biped. A set of fully-constrained stepping stone problems is solved using one, two, and three-step planning horizons. The distribution of stepping errors is shown. A one-step planning horizon performs poorly as it aims to accurately achieve the next foot placement while disregarding subsequent steps. The three-step planning horizon yields results that are qualitatively similar to the two-step planner, having slightly fewer large errors and fewer small errors. We hypothesize that the limitation on the quality of predictions made three steps into the future may be too low to yield an advantage over a two-step time horizon plan. A two-step time horizon further seems to allow sufficient flexibility for the posed problems.

Effect of sampling density during planning: The quality of the motion is a function of the number of samples used per state during the planning process, as shown in Figure 3.11. We collect stepping-length error data for stepping-stone sequences as a function of the number of samples used by the planner. Planning with regular sampling is used for this test. The solution quality improves as a function of the number of samples used.

Terrain stress test: Terrains can vary in difficulty and this can affect the ability of our simulated walking to successfully traverse it. A systematic characterization requires defining classes of terrain. As a simple test we develop a set of regular terrains that have a set of 5 gaps of width w and an inter-gap spacing of length s. We then record data for 3 simulated walks across the terrains, with each of these walks beginning at 3 different random distances from the first gap. We also need to

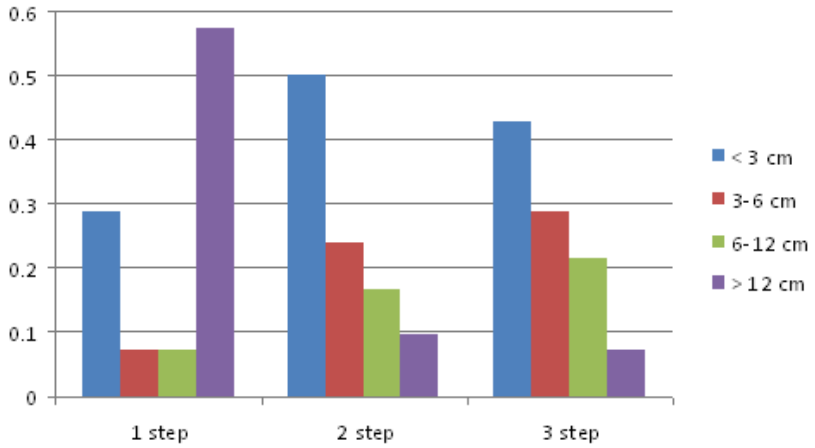


Figure 3.10: Effect of varying the planning horizon for a stepping-stone problem. The colored histograms give the distributions of stepping length errors.

		w			
		0.2	0.3	0.4	0.5
s	1	1	1	1	0.93
	0.75	1	1	1	0.93
	0.5	0.86	0.93	0.86	0.6
	0.25	0.8	0.93	0.66	0.3

Table 3.1: Effect of gap width and gap spacing on successful traversal. The fraction of successful steps is given for terrains with gap width w and inter-gap spacing s, as measured in metres.

define success. A successful walk can sometimes be obtained even without a solid foot placement on the far side of a gap. We define any footfall where less than half of the foot is on the ground to be a failure even if it does not result in a fall. Errors significant enough to cause a fall also count as a failure. Table 3.1 shows the results for various values of w and s and evaluated for the planar human model. As might be expected, the harder terrains are the ones with wider gaps and less space between the gaps. We note that traversing a $50cm$ gap requires taking a $70cm$ step, as measured heel-to-heel, and that our character has $90cm$ legs.

3.7 Discussion

The demonstrated technique shares ideas with kinematic data-driven methods such as motion graphs and their many variants. It is perhaps most similar with methods that develop continuously-parameterized kinematic models of motion. However, our work differs in several key respects.

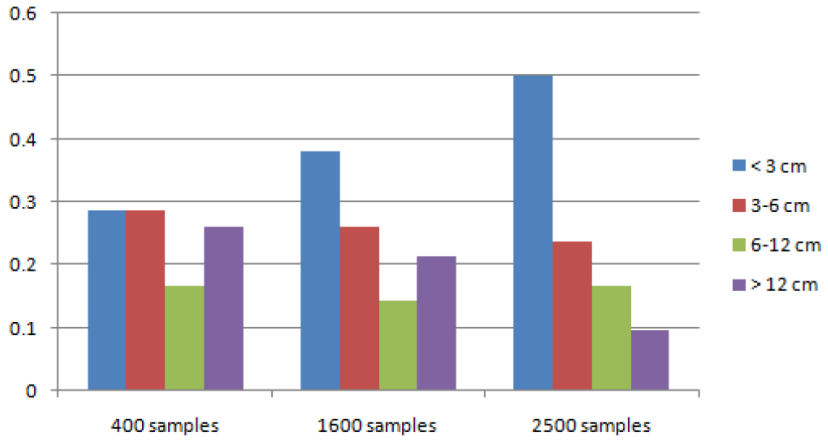


Figure 3.11: Effect of the number of interpolated sample actions used for exploring the action space during planning, applied to the stepping-stone problems of the type shown in Figure 3.7. The colored histograms give the distributions of stepping length errors. Regular sampling planning is used.

First, the model synthesizes its own example data to work from, which allows the method to work in the absence of motion capture data. Much of the power of computer graphics as a medium has always been its ability to portray new worlds and this requires abstract models that do not rely on large quantities of real-world data. The motion models developed in this chapter are the product of the physical structure of the given biped, the initial locomotion controller, physics, and the optimization objective function used to compute the offline example steps.

Second, while our approach is data-driven, it is applied to compute control actions that drive physics-based simulations instead of the kinematic interpolation of motions. Many kinematic techniques assume that it is possible to blend or transition between all pairs of stepping motions. We note that the analogous result does not hold in the dynamic setting. For example, a fast long step is infeasible without sufficient initial momentum. Applying a planning strategy analogous to that of motion graphs can be trivially implemented in our setting by considering the set of all stepping actions that begin from a state that is 'close enough' to the current state. In our framework this amounts to considering only the K actions beginning from similar states and which we use to define our action-space bounding box. We have experimented with this type of discrete action space and found it to be consistently inadequate. This motivates the sampling in a continuous action space that is used by our planner, which effectively allows for interpolation between previously observed actions.

Third, it is not obvious how to parameterize the dynamics of the example step data because it involves high-dimensional actions that govern transitions between high-dimensional states. Specific states and actions are unlikely to repeat. The 2D action space manifold, which we parameterize

using the first two PCA coordinates of the high-dimensional actions, introduces the necessary structure that makes sampling the action space a tractable proposition.

The results demonstrate that skills which anticipate features of the environment can be developed for real-time, reactive physics-based character animation in a largely automated way. Taken as a whole, the synthesis-analysis-synthesis process aims to automatically create a complete interconnected family of motions rather than individual motions. It establishes close connections between the constraints and objectives that shape a skill and the resulting patterns of action. The technique could likely be extended to other problems such as stepping over objects by using continuation methods to develop the required solutions to the example problems [Yin et al., 2008].

An interesting alternative to the current planning approach is to directly use regression to compute the next desired action. First, a desired sequence of target foot placements can be constructed using a simple fixed model of the minimum and maximum step lengths that the character can take. The action required for any given step could then be predicted directly from the example data set using regression, i.e., $A = q(s, l_1)$ or $A = q(s, l_1, l_2)$, where s is the current state of the character, l_1, l_2 are the next two desired step lengths, and q defines an appropriate regression-based estimator. Experimentation with this scheme revealed a number of limitations. One issue is that the planner needs to be very conservative in its placement of planned steps. Simply assuming that all step sequences satisfying the minimum and maximum step-length bounds of the example problems are equally feasible results in poor performance. A second issue is that it is not obvious how many future steps should be included in estimating the current action. Only considering the imminent step provides insufficient anticipation of upcoming steps. Considering the next two or three steps results in regression queries that have sparse (poor) data support, given that it is unlikely that the example data will contain a sufficiently similar example.

Our work has a number of limitations. The motions synthesized for our human-like 2D and 3D bipeds are not as natural as we would like. In particular, the 2D motion does not make significant use of the ankles and thus does not achieve the toe-off behavior expected in a human gait. Motion capture data could be used in two ways to help correct this. The base cyclic motion could be designed to more accurately mimic motion capture data. Additionally, if stepping sequence motion data were available, similarity to this data could be incorporated into the objective function for the offline synthesis.

The current method focusses on modeling the dynamics of variable length steps as seen in the sagittal-plane. This is sufficient for making the 3D model step across large gaps in the way that humans commonly do, namely stepping forwards across gaps and not sideways. It can be applied to general 3D curved paths as long as the required steps still happen predominantly in the sagittal

plane. However, this does not solve the fully general version of the stepping stone problem, namely navigating across an arbitrarily-placed sequence of 3D stepping stones, also perhaps having variations in height. Significant progress has been demonstrated on developing kinematic solutions to this class of problem [Choi et al., 2003; Safonova and Hodgins, 2007], although to the best of our knowledge these techniques do not yet, subjectively speaking, exhibit highly agile stepping and turning behaviors that would be indistinguishable from human motion captured directly in the same context. Extending our current technique to allow for diagonal steps would require adding one or two dimensions to the action space and adding extra dimensions to the low-dimensional state representation. We feel the technique would probably scale to accommodate diagonal steps with a small lateral component, although we have not tested such a scenario. Developing control strategies for much more arbitrary highly agile motions in highly constrained 3D environments remains an open problem, although we hope that our work may serve as a significant building block for this class of problem.

Chapter 4

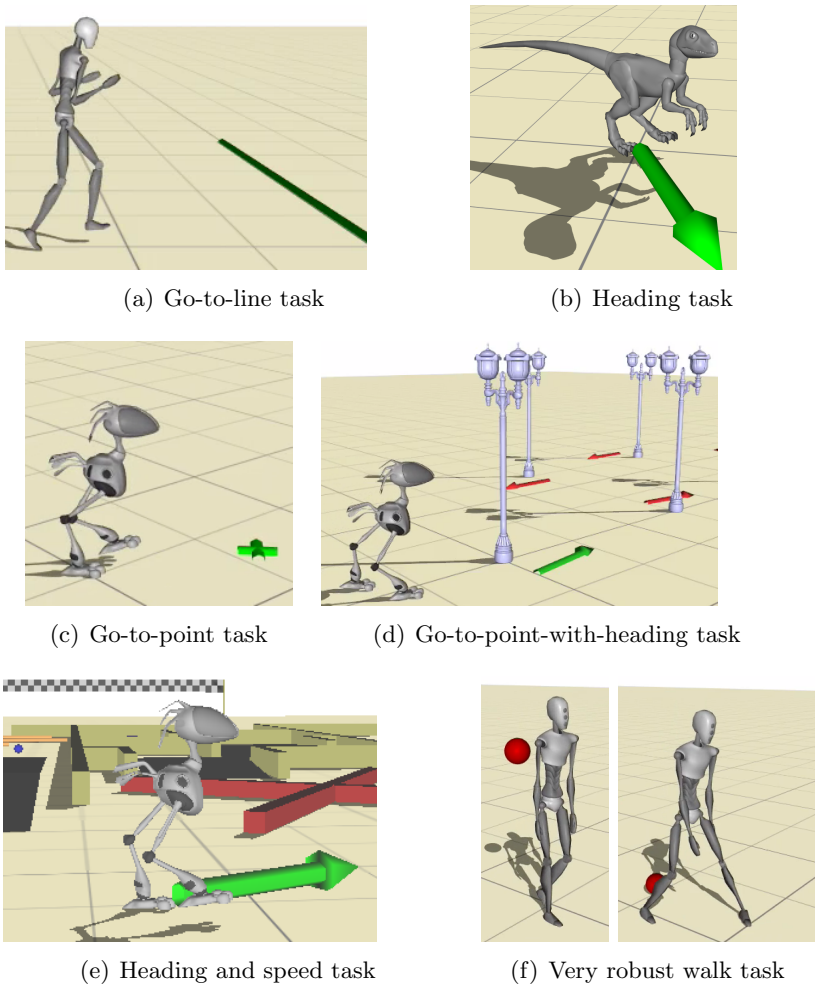
Task-level Control

In this chapter we investigate the problem of developing control policies that allow our characters to complete locomotion-based tasks such as getting to a target location as quickly as possible. In contrast to the work presented in the previous chapter, the action space that the control policies deal with is discrete and provided as input. Because our goal is to give rise to motions that are as agile as possible, we pre-compute control policies that maximize rewards over an infinite planning horizon. This is a different approach than the one we considered for the previous chapter, where the planners employed a 1 or 2 step horizon.

4.1 Introduction

Modeling the control needed for natural, agile motions is difficult. It is not only necessary to model individual skills such as walking and running, but also to find good ways of seamlessly combining these skills in order to produce purposeful motion that achieves a given goal. This chapter presents a method for the integration of physics-based locomotion skills towards solving tasks such as efficiently moving to a target point or target heading. The individual skills or actions consist of the locomotion controllers described in Section 2.4, and are assumed to be given as input. Their integration in task-based policies is non-trivial because the individual actions do not directly specify the final motion, as with kinematic models. Instead, they control it indirectly by steering the evolution of the high-dimensional character state. Given the many possible character states and the indirect steering nature of the available actions, this leads to an inordinately large space of possible motions that is difficult to model. To this end, we describe a process for creating a compact, restricted model of the dynamics with the help of a set of *trusted states*. The dynamics model is then developed by beginning at the trusted states and computing the state closure under the set of available individual actions, subject to a further limited-distance constraint with respect to the trusted states.

The task is specified using a reward or cost function, such as the distance to a goal point. The optimized control policy is then computed using fitted value iteration, which is a model-based



(a) Go-to-line task (b) Heading task

(c) Go-to-point task (d) Go-to-point-with-heading task

(e) Heading and speed task (f) Very robust walk task

Figure 4.1: We precompute task-specific control policies for real-time physics-based characters. The character moves towards the current goal in an optimized fashion, responds interactively to changes of the goal, and can respond to significant physical interaction with the environment during the task.

reinforcement learning algorithm. Importantly, the control policy is defined over the cross product of the character state and the task state, both of which are continuously-valued in our case and which results in a very high dimensional domain for the control policy. We show the feasibility of modeling control policies and value functions in such a case. The control policies also naturally integrate the need to maintain balance with the task objectives. For example, if a character is in a falling state and only one action can avoid a fall, this is captured by a control policy which is then invariant with respect to the task state for that particular character state.

Our computed control policies add robustness in a number of important ways. First, they exploit actions (controllers) where they are safe to use and avoid their use from states where this would lead

to failure. This allows for significant flexibility when designing the individual controllers because there is no need to explicitly supply a model of the preconditions for individual controllers. Second, although the form of controllers we use are already quite robust to external perturbations, we show that a task control policy makes them even more robust. Third, our results show that it is now possible to physically perturb simulated bipeds in significant ways while they continually and purposefully attempt to achieve their locomotion-based tasks. We are not aware of other demonstrations of this type of capability across multiple tasks and for multiple characters.

Figure 4.1 shows examples of tasks and characters for which we compute optimized control policies. The *go-to-line* task consists of walking (or running) forwards or backwards to the goal line. The goal of the *go-to-heading* task is to walk in a specified direction. The *go-to-point* task uses forward, backwards, or sideways walking to move to a goal location anywhere on the plane, and the *go-to-point-with-heading* task has the additional requirement that the character be facing in a given orientation when arriving at the goal. The *heading-and-speed* task walks in a specified direction at a specified speed. For all these tasks, the simulated character can respond in real time to changes of the goal and to physical interactions with the environment, such as stumbling over an object or responding to a push.

4.1.1 Overview

Figure 4.2 shows a block diagram of the system. The creation of task-based control policies begins with the design of the N *controllers* that will comprise the abstract action vocabulary available to the control policy. The task *control policy*, $\pi(s, w)$, then guides the motion on a step-by-step basis in a way that best achieves the task. Here, s is the character state and w is the task state, a simple example of which is the (x, y) position of the goal relative to the character, e.g., Figure 3.1(c).

The control policy for a task is pre-computed offline in two stages. First, a *state exploration* stage samples the dynamics of the character in a process that captures the evolution of the character state as different actions are applied. It also models the volume of state-space over which the control policy is to be defined. Given the sampled dynamics and a task description, the second stage computes an optimized control policy. This is accomplished by instancing the sampled dynamics into the task space and computing optimized actions for the cross-product of character states and task states. The control policy is built using *fitted value iteration*, which repeatedly improves upon a value function approximation (§4.4.3).

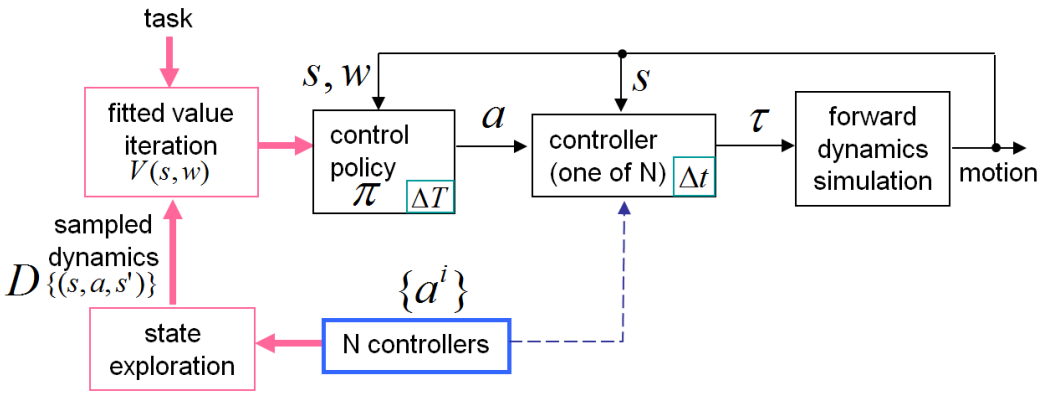


Figure 4.2: System overview. State exploration and fitted value iteration are performed offline, while the remaining blocks form the real-time control loop. The control policy makes decisions at every character step, ΔT, while the controller makes decisions at every simulation time step, Δt. w denotes the task state.

4.2 Previous Work

Control strategies have been developed for many physically-simulated motions, including hopping, running, vaulting, and bicycling [Raibert and Hodgins, 1991; Hodgins et al., 1995], standing balance [Khatib et al., 2004; Abe et al., 2007], falling and standing up after a fall [Zordan et al., 2005; Faloutsos et al., 2001], and walking [Laszlo et al., 1996; Sok et al., 2007; Yin et al., 2007; da Silva et al., 2008a; Muico et al., 2009]. Many of these control strategies can perform a variety of motions, such as walks of differing styles, and can demonstrate limited transitions between different controllers. Decisions regarding when to enact transitions between different controllers are generally left to the user or a supervisory controller with limited capabilities. Recent work has examined the composition of controllers using value-function-based interpolation [da Silva et al., 2009], instead of viewing it as a sequencing problem.

A number of methods have recently been proposed for task-based control policies using kinematic motion models. These specify the best motion clip to transition to, based on knowledge of the task at hand and the identity of the currently-playing clip [Choi et al., 2003; Lee and Lee, 2004; Lau and Kuffner, 2005; Ikemoto et al., 2005; Lau and Kuffner, 2006; Treuille et al., 2007; McCann and Pollard, 2007; Lo and Zwicker, 2008; Zhao and Safonova, 2008]. Objective functions for this kind of motion planning strike a compromise between the quality of transitions between motion clips (visible jumps are undesirable) and task-related criteria, such as the time or effort used to meet a desired goal. The methods are commonly applied to a graph-based model of possible motions that is instanced in the task space at chosen sample points. The control policy is computed in unison with a value function, which is a function of both the character state, s, as represented by the current

motion clip, as well as the task state, w. Given a finite number of motion clips, s is discrete in nature for most kinematic motion models. However, s is continuous when working with dynamic characters because the result of physical simulations is not fully constrained. As we shall detail later (§4.6), our controller-based actions behave differently from kinematic motion clips in a number of important respects. Most fundamentally, kinematic task control policies are limited in that they do not allow physical interaction with the surrounding world.

Reinforcement learning has been applied in robotics in order to develop control policies for walking [Atkeson and Stephens, 2007; Morimoto and Atkeson, 2007; Morimoto et al., 2007; Byl and Tedrake, 2008]. Much of this work focuses on developing optimal controllers for steady-state walking, while the primary focus of our work lies with higher-level tasks. Like much of this work, however, our method plans on a step-by-step basis, i.e., using Poincaré sections sampled at foot contact.

Several methods have been developed to enable real and simulated humanoid robots to do online planning for reaching goal locations while also avoiding known static and dynamic obstacles [Chestnutt, 2007; Yoshida et al., 2005]. Terrain navigation can be achieved by generating footstep plans. These are then given as input to walking control strategies which may be based on preview-control methods [Kajita et al., 2003] or proprietary methods, i.e., Honda ASIMO. The footstep plans can be generated using A* search [Chestnutt et al., 2005]. or with the help of precomputation [Lau and Kuffner, 2006]. However, walking motions for current biped humanoid robots remain quite fragile and demonstrations of robustness to moderate pushes are a recent development. One of our goals is to develop physics-based characters that can cope with significant physical interaction with the environment during the motion, this being one of the primary benefits of using physics-based simulation in applications such as games. The allowable footstep placements for footstep-based planning generally need to be conservatively designed, which helps limit the extent to which the robot state needs to be considered when planning the next step. We allow for controllers and controller transitions that can be quite dynamic but that can fail as a result. Coping with this is then the job of the task-based control policy. We forego the deliberate-and-precise foot placement that is common to humanoid robot control and demonstrate an alternative approach for achieving locomotion tasks in a robust and purposeful fashion.

4.3 Actions

Developing compact models for the action space is crucial for scaling reinforcement learning to high-dimensional settings. For this reason we work with a discrete action space and make decisions at the granularity of character steps. Each action consists of a low-level locomotion controller, such as one step of a walk or run cycle.

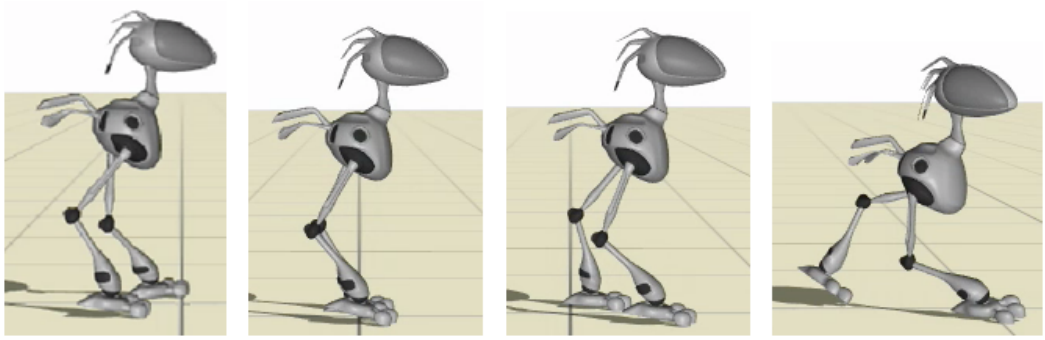


Figure 4.3: The outcome of an action is highly state dependent in our framework. This example shows the results of the same action applied to four different initial states.

For our implementation, we use locomotion controllers of the type proposed by [Yin et al., 2007]. These controllers track target joint trajectories using proportional-derivative control. Balance strategies are incorporated by having the torso and swing hip track desired trajectories in a world frame, and also by using continuous feedback that adjusts the placement of the swing foot. The action vocabulary consists of either predesigned controllers, or intermediate controllers, which are defined by interpolating the torques output by two other controllers. We later provide guidelines for the design of controllers to be used in solving a task (§4.5).

While the controllers are constructed with the help of a target motion, they generally do not rapidly bring the character state onto a specific motion trajectory. As such, they behave in ways that are qualitatively different from high-gain trajectory tracking controllers. For example, the same controller can be exploited for different purposes in different situations. A forwards walking controller that is invoked from a backwards walking state may be used to induce a rapid stop, while the same motion invoked from an in-place walk will induce forwards motion. This is illustrated in Figure 4.3. In such situations, the controller is best abstractly characterized as implementing an acceleration action, and not as tighly tracking a specific target motion. The impact of the controller behavior on the modeling of the dynamics is further examined in §4.6.

4.4 Policy Synthesis

Given a vocabulary of actions $A : \{a^i\}$, the control policy needs to decide which action should be used every time the character takes a step. It does this as a function of the continuously-valued character state, $s \in \mathbb{S}$, and task state, $w \in \mathbb{W}$. The character state s consists of the positions and velocities of the degrees of freedom of the character as seen in its own coordinate frame. Formally, $s = (p, v, q_0, \omega_0, q_1, \omega_1, ..., q_n, \omega_n)$, where p, v, q_0, ω_0 are the root position, velocity, orientation and

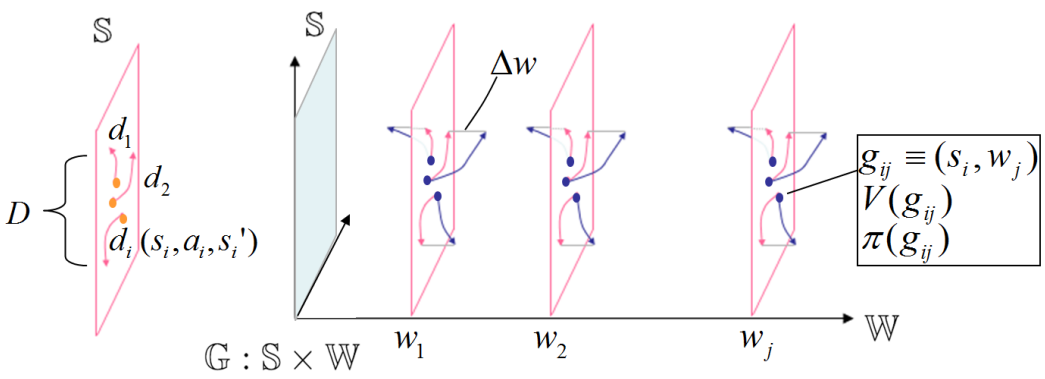


Figure 4.4: Schematic illustration of how sampled state-to-state dynamics are instanced in the task space. Left: Example transitions as seen in state-space, $\mathbb{S}$. Right: Example transitions as seen in $\mathbb{S} \times \mathbb{W}$. The out of plane state-transition arcs, Δw, are computed from s and s'. w refers to the task state.

angular velocity, and q_i, ω_i are the relative orientation and angular velocity of joint i. The orientations are represented by quaternions. The task state w is used to parameterize the task. For example, it could be used to represent the relative position and orientation of the goal with respect to the character. We also define a *global state* $g = (s, w)$ and its corresponding space $\mathbb{G} : \mathbb{S} \times \mathbb{W}$. Formally, the control policy is defined as a mapping $\pi : \mathbb{G} \rightarrow A$, or, equivalently, $a = \pi(g)$.

Policy synthesis begins by modeling the dynamics of the character state in the *state exploration* stage. It is impractical to create a sample-based model for the dynamics that encompasses all possible states because of the high-dimensional nature of the character state. Instead, we focus on modeling the dynamics in a limited region of interest. This is defined with the help of a set of *trusted states* which are treated as starting points for a structured, constrained exploration of the state space. We use a tree-structured exploration process using the set of available actions. This computes an approximate closure for the trusted states under the set of available actions. The character dynamics is modeled as the state transitions encountered during this process, which are captured as a set of dynamics tuples, $D : \{(s, a, s')\}$. Each tuple represents one character step in the context of our locomotion-based tasks, and records the starting state of the character s, the applied action a and the resulting state s'. We find s' by running a forward-dynamics simulation using the low-level controller associated with a until the next foot-strike. An abstraction of the recorded state-to-state dynamics is shown in Figure 4.4 (left).

Given the dynamics model, the control policy is computed by instancing the character dynamics at multiple sample points in the task space and applying an optimization procedure, *fitted value iteration*, to compute the best global policy. We now describe state exploration, instancing the character dynamics, and fitted value iteration in more detail.

4.4.1 State Exploration

State exploration samples the character dynamics by beginning at a given set of trusted states T and recursively exploring the application of all possible actions, as described in Algorithm 1. With each iteration we grow the set of sampled character dynamics tuples, D, by adding new state-action-result tuples (s, a, s'), if they meet two criteria. First, the resulting state s' needs to remain sufficiently close to one of the trusted states. This constraint serves as a simple way of modeling what natural poses during a motion are expected to look like and it helps focus our resources on regions of state space that are likely to be important. Second, further recursion is rejected if s' is not considered to be a novel state, i.e., it is within a threshold distance of an existing state. This prevents revisiting an already-explored area. The *trusted* and *novelState* functions both make use of the same character-specific state distance metric, $\mathrm{d}(s_a, s_b)$. For state s' to be considered trustworthy requires $\mathrm{d}(s', s) < \epsilon_T$ for at least one state $s \in T$. A novel state s' is defined as one which satisifies $\mathrm{d}(s', s) > \epsilon_N$ for all states $s \in D$. The character state distance metric is defined by:

$$\begin{aligned}\mathrm{d}(s_a, s_b) = w_v|v_{s_a} - v_{s_b}| + \sum_{i=0}^{n} w_{\omega_i}|\omega_{i,s_a} - \omega_{i,s_b}| \\ + \sum_{i=1}^{n} w_{q_i}\mathrm{dq}(q_{i,s_a}, q_{i,s_b}),\end{aligned}$$

where $\mathrm{dq}(q_a, q_b)$ is the rotation angle, in radians, represented by the quaternion $q_a^{-1}q_b$. The weights used are character specific and are defined in Table 4.1.

4.4.2 Instancing Dynamics

Figure 4.4 (right) provides an abstract illustration of how the sampled dynamics is conceptually instanced at multiple points in the task space to produce a dynamics model that spans both the character state and the task state. As with motion graphs, the dynamics of the character state is treated as being invariant with respect to the (x, z) position (where y is up) and the facing orientation of the character in the world. We adopt an approach similar to that of [Treuille et al., 2007] and [Lo and Zwicker, 2008] and use a set of task state sample points, $\{w_j\}$ that cover the space of possible goal positions and/or orientations as required by specific tasks. During instancing, the effect of each action on the task state, Δw, is computed from s and s' for every dynamics tuple d, as illustrated in Figure 4.4. The result of this operation is a set of augmented dynamics tuples $(s, w, a, s', w') \equiv (g, a, g')$.

Algorithm 1 State Exploration

```
1: input Q: queue of states
2: input A = {a^i}: set of actions
3: input T: set of trusted states
4: output D = {(s, a, s')}: set of dynamics transition tuples
5: enqueue(s, ∀s ∈ T)
6: while s ← dequeue() do
7:   for all a^i ∈ A do
8:     s' = forward_dynamics_simulation(s, a^i)
9:     if trusted(s') then
10:      D = D ∪ {(s, a^i, s')}
11:      if novelState(s') then
12:        enqueue(s')
13:      end if
14:    end if
15:  end for
16: end while
```

4.4.3 Fitted Value Iteration

We compute the optimized control policy using a reinforcement learning framework [Sutton and Barto, 1998]. Tasks are specified via a reward function $R(g, a)$. The reward function measures the immediate benefit of the character taking action a when at state g. The goal of the optimal control policy is to maximize the cumulative long term reward, $V(g) = \sum_{t=0}^{\infty} \gamma^t R(g_t, a_t)$ where $\gamma \in (0, 1)$ is a discount factor that ensures a finite cumulative reward over an infinite planning horizon. The optimal value function can be written in the recursive form given by the Bellman equation,

$$V^*(g) = \max_a \big(R(g, a) + \gamma V^*(g')\big),$$

where $g' = (s', w')$ represents the character and task states which result from applying action a at g. For any given global state g the optimal policy, $\pi^*(g)$, is given by the action that maximizes $V^*(g)$.

Estimations of the value function v_{ij} are stored at each sample point g_{ij}, corresponding to the instancing of state s_i at the task state point w_j. After initializing all v_{ij} to zero, fitted value iteration (FVI) [Ernst et al., 2005] works by iteratively computing improved value function estimates for all g_{ij}. The resulting (g_{ij}, v_{ij}) tuples are then used to define the new value function $V(g)$ using locally-weighted interpolation. Algorithm 2 provides a full description of the process. We use a form of FVI that follows the core idea of the $TD(0)$ temporal difference learning method with learning rate α. We update the value function in-place, which simplifies our implementation, but we could equivalently use other variants of FVI such as batch-mode FVI [Ernst et al., 2005; Lo and Zwicker,

Algorithm 2 Fitted Value Iteration

input $A = \{a^i\}$: set of abstract actions
input $D = \{(g, a, g')\}$: global state dynamics tuples
input $R(g, a)$: reward function
output $\pi^*(g)$: control policy over $\mathbb{S} \times \mathbb{W}$
output $V(g)$: value function over $\mathbb{S} \times \mathbb{W}$
$V(g) = 0$ for all $g \in \mathbb{G}$
while not converged **do**
  for all $g \in \mathbb{G}$ **do**
    $a^* = \underset{a \in A}{\operatorname{argmax}}\big(R(g, a) + \gamma V(g')\big)$
    $\pi^*(g) = a^*$
    $\tilde{V}(g) = R(g, a^*) + \gamma V(g')$
    $V(g) \leftarrow \alpha \tilde{V}(g) + (1 - \alpha) V(g)$
  end for
end while

2008]. We have experimented with varying the learning rate, α, and the resulting policy is largely unaffected although overly small values of α can dramatically slow the convergence rate. In practice, we use $\alpha = 0.8$.

Due to the step-to-step nature of our planning process, by default the discounting of the rewards would be step-based rather than time-based. As a result, the character aims to reach the goal in the fewest number of steps rather than in the shortest amount of time. We can alter this behavior by making the discount rate be a function of ΔT, the duration of a locomotion step. To this end, we use $\gamma(\Delta T) = \gamma^{\Delta T}$.

Given a query point $g_q = (s_q, w_q)$ we estimate $V(g_q)$ from a set of sample points, $\{(g_{ij}, v_{ij})\}$, using a mix of kNN and multilinear interpolation. We first identify the set of k nearest neighbors $\{\hat{s}_i\}$, among all the sampled character states. For each $\hat{s}_i$, we use multilinear interpolation to obtain an estimate $V(\hat{s}_i, w_q)$. Finally, we perform kNN interpolation among these values to obtain $V(g_q) = \sum_i f(\mathrm{d}(\hat{s}_i, s_q))V(\hat{s}_i, w_q)$. In the abstract view shown in Figure 4.4, this corresponds to first interpolating onto the plane corresponding to w_q for all k neighboring states in $\mathbb{S}$, and then interpolating within this new plane for the query character state, s_q, using kNN interpolation. We use $k = 6$ and a weighting kernel defined by $f = 1/\mathrm{d}^2$. Since FVI repeatedly needs to evaluate the value function at the states s' found in D, we precompute and store their k nearest neighbors.

In addition to storing value function estimates, we also store the optimal action $\pi^*(g_{ij})$, computed for each sample point g_{ij}. At run time, for a query point $g_q = (s_q, w_q)$, the control policy selects the action associated with g_{ij} where s_i is the character state closest to s_q and w_j is the task state closest to w_q. We use this form of nearest-neighbors rather than a weighted interpolation when

	humanoid		bird		raptor	
	w_ω	w_q	w_ω	w_q	w_ω	w_q
torso-neck	—		1	1.5	1	1.5
pelvis-torso	1	1.5	—		—	
hips	0.5	1.5	0.1	1.5	0.1	1.5
knees	0.2	1	0.05	1	0.05	1
neck-head	0	0	0.05	1.5	0.05	1.5
body-tail	—		—		0.05	1

Table 4.1: Character-specific weights used in the state distance metric. The weights for all other joints are zero.

computing the desired actions in order to remain consistent with a discrete action model. However, given compatible action representations, we speculate that interpolation would also likely produce good results.

4.5 Results

Implementation: For simulations we use the Open Dynamics Engine [ODE, 2010] on a 2.4 GHz Intel Core 2 Duo with a simulation time step of 0.0005 s. All of our results run faster than real time, meaning that in one wall-clock second we can advance the simulation time by more than one second: 2.5× for the bird character, 1.5× for the humanoid and 1.1× for the raptor. The humanoid character has a total mass of 70.4 kg and has 34 degrees of freedom. The bird character has a total mass of 64.1 kg and a total of 24 degrees of freedom. The raptor character has a total mass of 77.8 kg and 42 degrees of freedom, many of which are in the tail. The balls thrown at the characters have a mass of 7—15 kg. The weights for the character-specific distance metric are listed in Table 4.1 and are set with the help of knowing the mass distribution of the character. For example, the big bird has a heavy head relative to the rest of its body and light-weight legs, so the weights used for the related joints reflect this. The head of the humanoid is small and unlikely to have much of an influence on the overall dynamics of the character and thus we set the weights on the related joints to zero in this case.

Controller design:

Individual controllers of the type described in Section 2.4 are used as abstract actions. Designing basic low-level controllers is not hard given the right tools and a basic in-place stepping or walking controller to use as a starting point. We have had new users design good running controllers in less than an hour in this way. However, it takes more time to design very natural looking motions and these are often less robust. The final control policy will in the end only be as natural as the

	ϵ_T	ϵ_N	$\|T\|$	$\|s\|$	FVI time
Bird GLT	3.75	0.15	7	2494	100 s
Humanoid GLT	2.0	0.3	24	2253	104 s
Bird HT	1.5	0.3	16	1915	92 s
Raptor HT	2.0	0.2	60	2177	86 s
Bird GPT	1.5	0.4	50	4375	5 h
Bird GPHT	1.5	0.3	21	1963	9.5 h
Humanoid VRW	6.0	1.5	30	1710	10 s

Table 4.2: Constants and numerical results for the various tasks. Here, $|s|$ indicates the number of states sampled during the state exploration stage using the specified values for ϵ_T and ϵ_N and a total of $|T|$ trusted states. The last column indicates the total time required for fitted value iteration.

underlying controllers. In designing the controllers, the general goal is to span the range of desired gaits needed for the task while also testing to ensure that a variety of reasonable transitions are possible between the controllers. The control policy synthesis method is demonstrably forgiving to limited-robustness controllers that only support transitions in limited situations (§4.6).

Trusted states: The trusted states are chosen to represent the steady-state operation of the basic actions, as well as to encompass natural-looking states obtained when transitioning between different locomotion modes. The set of trusted states is initialized by sampling the limit-cycle states for several of the actions, i.e., forward run, forward walk, in-place walk, backwards walk. Further trusted states are identified with the help of a small number of manually-generated action sequences that are simulated and then observed to see if they yield subjectively 'good' motions, i.e., no falls or other undesirable artifacts. We typically use five sequences of five actions each, which provides a coarse sampling of the kinds of motions that can be generated without needing to run long or exhaustive exploration. For each state s in a good sequence, we test to ensure that *trusted(s)* evaluates to *true*; if this is not the case, we add s to the set of trusted states.

Policy computation: Details pertaining to the various tasks are presented in Table 4.2. For all our experiments, the state exploration stage takes between 2.5 and 4 hours, and the FVI converges in 50–70 iterations. The control policies require 0.5—2.5 Mb to store.

4.5.1 Go-to-line Task (GLT)

The goal is to reach a designated line by walking forwards, running forwards, or walking backwards towards it. The task state is one-dimensional and represents the position of the line relative to the character. To compute the control policy, we use a total of 35 task state sample points, $-2.5m \leq t_j \leq 6m$, sampled more densely around the origin where extra precision is required for

stopping at the right point. The change in task state, Δw as shown in Figure 4.4, is given by the change in root position between s and s', projected onto the sagittal-plane.

The reward function for this task is simple. A reward of 1 is given when the character is within $10cm$ of the goal and has a near-zero forward velocity. A reward of -1 is given when the character is in a non-trusted state. In all other cases, the reward is 0. For the bird character, a set of 10 actions is used: backwards run, inplace walk, forward walk and forward run, as well as 6 intermediate actions obtained through interpolation. For the humanoid, we use a similar set of actions to which we add three actions that help improve the visual quality of the transitions between walks and run.

4.5.2 Heading Task (HT)

For this task, the character's goal is to be walking forward in a specified direction. The task state is one dimesional, consisting of the desired heading direction, θ, as measured relative to the current character heading. We compute the control policy over $-\pi < \theta \leq \pi$, and sample at 25 points in this range, sampled more densely around zero. The difference in heading directions between s and s' defines Δw.

The reward function is again very simple. If the character is walking forward at steady state within 5 degrees of the specified direction, the reward is 1. If the character state is outside the trust region the reward is -1, and otherwise it is 0. A set of 32 actions is used by the bird character. To develop these, we designed an in-place walk, a forward walk and two clock-wise turning controllers. The turning controllers are mirrored to achieve counter-clockwise turning. The rest of the actions represent various intermediate controllers. A similar set of actions was used for the raptor controllers. However, we use fewer intermediate controllers resulting in a total of 15 actions.

4.5.3 Go-to-point Task (GPT)

A more complex task involves the bird character moving to a specified goal location anywhere on the ground plane. A 2-dimensional task state, (x, z), is used to represent the location of the target relative to the character's frame of reference. We use a total of 729 task states sampled along a $6m \times 6m$ axis-aligned grid with spatially varying density to allow for denser sampling near the goal location, $(0, 0)$. Δw is set to the ground-projected difference in root positions between s and s'.

We augment the set of actions for the heading task with a backwards walk, a side-stepping motion (mirrored left and right) and one additional interpolated action. This yields a total of 36 actions. Similar to the go-to-line task, the reward function returns 1 when the character is walking in place

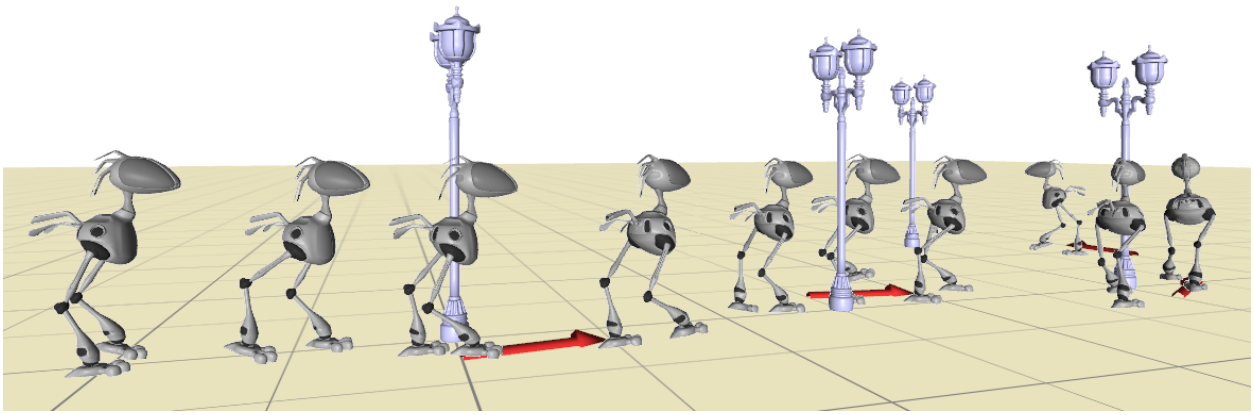


Figure 4.5: Smoothly walking around lamp posts by walking to goal points with headings.

within 10 cm of the goal, -1 when the character's state is too far from the trusted states and 0 otherwise. This simple reward scheme reveals an unexpected behavior: the bird character sometimes turns and sidesteps in order to quickly stop at the target. To eliminate this behavior, we add a term to the reward function that penalizes large changes in orientation when within $1.5m$ of the target. This demonstrates a simple use of the reward function to help shape the solutions produced by the control policy.

4.5.4 Go-to-point-with-heading Task (GPHT)

This task combines the go-to-point and heading tasks. Here, the goal is to be walking at steady-state through a target position while facing a specified orientation. This makes it particularly useful for navigation in environments, as smooth motion paths can be specified using a sparse number of such goals that act as way-points. Specifying a desired heading can also be used to eliminate the unexpected behavior experienced in the go-to-point task. Figure 4.5 shows a resulting motion.

For this task, we use a three-dimensional task state, (x, z, θ). The positions are sampled using a 16×16 grid spanning a $6m \times 6m$ area, and 8 samples are used to sample the orientation dimension, for a total of 2048 task state sample points. The sampling is denser around the goal position and orientation. The value of Δw is obtained by combining the analogous components described for the two previous tasks. We use a total of 36 actions, which include those of the heading task, augmented by a backwards walk, left-and-right side-stepping motions, and one additional interpolated action. The reward function is 1 when the goal is satisfied, -1 when the state is non-trusted, and 0 otherwise.

4.5.5 Heading with Speed Task (HST)

We build a control policy for following a given heading at a given speed by building on the results of the state exploration for the heading task. The task space is sampled using $5 \times 25 = 125$ states, corresponding to the speed and heading directions, respectively. States that approximately satisfy the desired speed in the heading task are rewarded. The supplemental material includes an interactive game-like demonstration where the player can steer a character through a dynamic, obstacle-filled environment by controlling the character using the desired heading and speed.

4.5.6 Very Robust Walk Task (VRWT)

We generate a particularly robust steady-state walking gait for the humanoid by adding a set of trusted states that correspond to the state at the next foot contact after being hit by $10kg$ and $15kg$ balls. The task is simply to walk at steady state, and so there is no task sampling. The reward is 1 for being at (or near to) the steady state, and 0 otherwise. For actions, we use the in-place walk, forward walk, backwards walk, two modified inplace walks (one leaning slightly forwards and another leaning slightly backwards), and several other interpolated controllers, for a total of 13 actions.

We compare the resulting task control policy to the forward walk controller. For $5kg$ balls thrown at the character at different points in the walk cycle and from random directions, the failure rate with the computed policy drops from approximately 30% to 0%. For $10kg$ and $15kg$ balls, the change in failure rate is similarly $70\% \rightarrow 10\%$ and $100\% \rightarrow 10\%$, respectively.

4.6 Discussion

Computed vs hand-designed policies: It is informative to compare the computed policies to carefully hand-crafted policies. For tasks such as the three-dimensional GPH task, hand-designing a good control policy is impractical because of the complexity. Thus, a first benefit of the computed policies is their ease of design using a simple reward function. Even for simpler tasks, optimized policies are not easy to approximate by hand. Figure 4.6 visualizes a computed control policy for some of the states from the humanoid go-to-line task. The best actions are both character-state dependent and task dependent. Another difficulty of hand-crafting policies is that the outcome of an action is dependent on the initial state. For comparison, we also develop a carefully hand-tuned controller for the bird heading-with-speed task and quantitatively compare its mean performance to

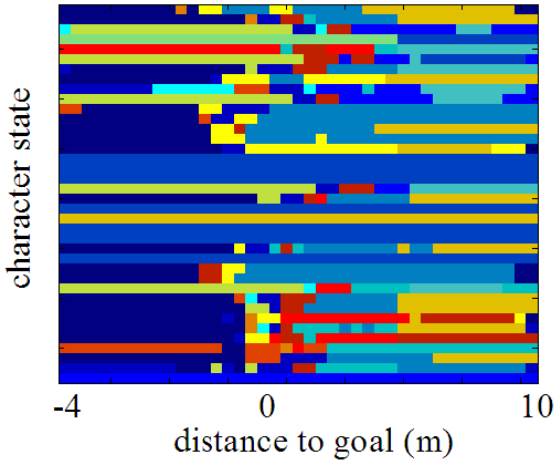


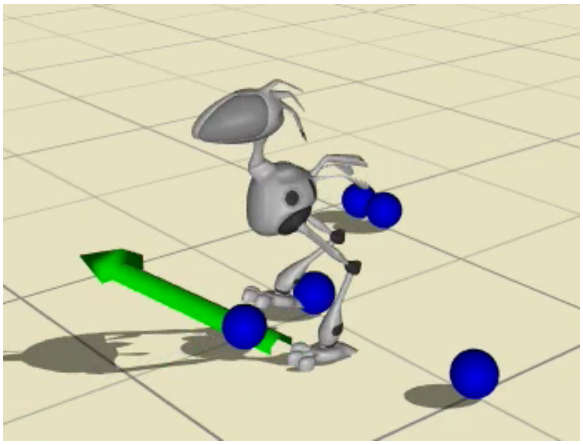
Figure 4.6: Partial visualization of the humanoid go-to-line control policy. The rows correspond to a random selection of character states. The columns correspond to different distances of the character to the line. Actions are assigned an arbitrary color.

that of the optimized policy. Both situations are evaluated from a regularly sampled set of initial task states. The optimized control policy outperforms the hand-designed policy by up to 32% for some states and a mean difference of 17 % in terms of time-to-goal. This is computed over 96 initial states and ignores the cases where the hand-tuned controller fails outright. Optimized control policies also avoid failure-prone transitions between controllers, as we discuss next.

Robustness: A primary motivation for physics-based character animation is that the motions can respond in meaningful ways to a variety of physical interactions with the environment. We perturb the character's motions with heavy balls and a variety of objects that can be stepped on or tripped over, examples of which are shown in Figure 4.7. We are not aware of other demonstrations of simulated bipedal characters or humanoid robots that have this type of robustness during task execution.

While the underlying controllers are already robust to a certain extent, the task control policy adds robustness in two respects. First, task policies anticipate and avoid the application of actions that lead into areas of the character state space where failure is inevitable. The controllers themselves provide no safeguards with respect to being invoked in an inappropriate state. A simple prototypical case is that of the turning controllers used for the raptor heading task which are quite sensitive to the initial character state and can result in an awkward side-stepping behaviour leading to an eventual fall.

The computed control policies are also more robust to external perturbations than the individual controllers. For example, a walking controller can receive a perturbation, which, if the walking



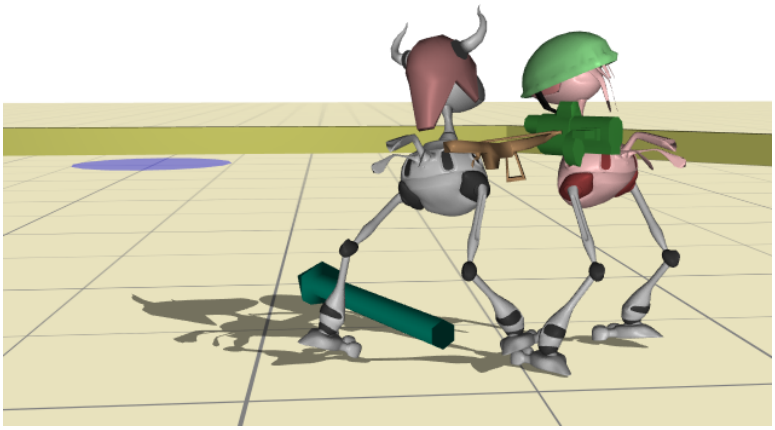


Figure 4.7: Interactions with the environment or with other characters can happen at any time during the task.

controller remained active for the subsequent steps would lead to failure. With the use of a task-based conrol policy, a similar state may have been seen during the state exploration phase and as a result, the control policy is well adapted to recover from this state if this can be achieved through the application of another action. Put simply, the task policy combines the strengths of the individual controllers. We note that while the control policies act on a step-by-step basis, the recovery from a perturbtation does not need to happen in a single step. The key is that the character state at any point should never stray too far from example states that have been observed during the state exploration phase.

Graph-based models of motion: Control policies can be simpler to develop if the dynamics of the motion can be modeled using a motion graph, wherein there are a discrete set of character states connected by particular available motions. With such a motion model, the character state is assumed to evolve in a predictable and highly constrained way, as given by the discrete choice of paths through the graph. A graph abstraction is used by recent developments that build kinematic control policies based on step-based motion clips [Treuille et al., 2007; Lo and Zwicker, 2008]. A blending model is used to allow for transitions between all pairs of motion clips in any given step. In effect, this acts as a funnel that always achieves the end-state of the applied action in one step, irrespective of the starting state. The number of actions and the number of possible character states are equal to the number of motion clips. A related dynamics model is also developed for the Honda ASIMO robot in [Chestnutt et al., 2005], where the history of the last two actions is used as a proxy for the dynamic state of the robot. When used with a set of 7 discrete actions, this yields a motion graph with 49 unique states and 49×7 edges. The actions thus act as a funnel that achieves a fully predictable end-state after two steps.

The state exploration phase of our method can also be thought of as forming a type of motion graph if we consider state-exploration process as reconnecting to existing motions whenever it passes sufficiently close. The *novelState* function as used in Algorithm 1 effectively models this condition when it is used to help prevent revisiting explored areas of state space. However, this graph has significantly different properties from the graphs described above. The humanoid go-to-line task uses 16 controllers and the exploration yields a total of 4806 unique states, which is significantly larger than for the graph-based methods described above. The number of viable actions per state ranges from 0—14, with a mean of 8 viable actions. This stands in contrast to previous graph-based methods, which generally assume that all actions are viable in all states. The humanoid go-to-line task requires a mean of 6 successive actions to begin at a trusted state and then either fail the trusted-state condition or to approximately reach a repeating state. This contrasts sharply with previous graph-based models of motion which assume that the state history can be fully forgotten after only one or two successive actions. In summary, the actions used in our framework do not behave in the same way that is assumed in prior work on kinematic control policies.

A last notable difference between our work and that of graph-based dynamical models is that our policy and underlying value function are defined over a continuous character state-space, which is not true of graph-based models. The control actions at each step are computed exclusively as a function of the actual current character state. In this respect the method differs from classical tracking approaches; there is no notion of a continuous target trajectory that is always being tracked. A trajectory tracking model needs to know when to stop tracking a given trajectory and jump to a new trajectory, an idea that is explored in part in [Sok et al., 2007]. In this sense, the step-by-step decisions of the task control policy can be seen as jumping to a new trajectory or controller at every step. We speculate that less-robust low-level controllers could be used in our framework, but at the expense of having the high-level control policy do more of the work. As noted earlier, both the high-level control policy and the low-level controllers contribute to the final robustness of our controllers.

Dimensionality: A core challenge in applying reinforcement learning techniques to physics-based characters is the high-dimensional nature of the value function, which needs to span the cross-product of the character state and the task state. We currently rely on two important pieces of *a priori* knowledge to deal with the high-dimension of the state space. Some knowledge of the expected motions is used to seed the state space with a set of trusted states, and thereby help focus the use of resources on modeling the dynamics and control in task-relevant regions of the state space. We also define a character-state distance metric that is used at several points in our pipeline. Good distance metrics can provide meaningful interpolation behavior while avoiding some of the complications that are associated with explicitly modeling low-dimensional embeddings. In future work it would be interesting to automatically learn the best distance metric to use.

Multiple levels of abstraction also contribute towards tackling the high-dimensional nature of the control policies. For example, four levels of abstraction are used in the bird-mania game: (1) the player commands a desired heading-and-speed; (2) the heading-and-speed task policy commands an individual controller; (3) an individual controller commands a set of joint target angles; and (4) joint PD-controllers command joint torques. When seen as a whole, these layers of control implement a mapping from desired heading-and-speed to joint torques. However, directly learning such a mapping is much more difficult than learning level 2 alone, i.e., the task policy.

Scalability: A number of factors affect the run-time complexity and storage costs of the method: the dimensionality and sampling of the task space, the size of the action vocabulary, the maximum trusted state distance (ϵ_T), and the minimum novel state distance (ϵ_N). The dimensionality of the task space is perhaps the most important – the $2D$ and $3D$ tasks (GPT, HST, GPHT) require hours to compute the final policy, as compared to the minutes required for the $1D$ tasks. This is because the number of task-space sample states grows exponentially with the task space dimension.

$\|s\|$	$\|w\|$				
	81	57	32	25	17
4806	0	0.22	0.22	0.23	0.34
4000	0.18	0.35	0.28	0.25	0.35
3000	0.45	0.58	0.54	0.42	0.40
2000	0.69	0.86	0.80	0.65	0.50
1000	0.79	0.99	0.87	0.75	0.60

Table 4.3: Effect of character state sampling and task state sampling for the humanoid go-to-line task. The table gives the mean value function error and uses the most densely sampled case as a baseline.

We speculate that many tasks which are nominally high dimensional can often still be tackled using lower-dimensional policies. For example, a thrown ball may have a 6D state (3D position and velocity) relative to a character, but can be caught by running quickly to a good intercept point. In this way, additional task and action abstraction may mitigate the complexity of apparently high-dimensional tasks.

The complexity of the character state dynamics does not vary significantly across our examples. All use an action vocabulary consisting of 10–36 controllers and the resulting state dynamics is modeled using 1700–4800 sampled states. This is partly by design – if the set of trusted states and choice of ϵ_T allow for an expansive exploration of irrelevant regions of the state space, then modeling the state dynamics can become prohibitively expensive.

Parameter settings: In order to better understand the sensitivity of the computed task policies to the character state sampling and the task state sampling, we look at how the task policy changes as a function of the sampling for the humanoid go-to-line task. The results are given in Table 4.3. The quality of the solution generally worsens as we decrease the state sampling and the task state sampling. We have also evaluated the quality by measuring the percentage of actions that remain the same as the optimal choice of action (assumed to be given by the highest sampling) as the sampling is decreased. This decreases monotonically as we decrease either the state sampling or the task state sampling. Sparse sampling may also result in discrepancies between the modeled value function and the rewards encountered during actual policy execution.

As another test, we varied the number of actions used for the same task. We use the same baseline case, i.e., $|s| = 4806$, $|w| = 81$ and which uses 13 actions. For $\{10, 7, 4\}$ actions, the average increase in the value function error across all sampled states is given by $\{1.58, 3.31, 5.46\}$. Thus, a richer action vocabulary clearly does help in better achieving the task.

Limitations: The current method has a number of limitations. For any given task, the discrete set of abstract control actions is restrictive as compared to the more arbitrary actions that are

afforded by the low-level dynamics. The control actions also remain step-based, which precludes making task-based adaptations to a motion halfway through a step. As a result of such issues, the motions are still not as agile as we would like them to be. It is not always obvious what set of controllers should be made available to the control policy as abstract actions for a given task. The time required for the state exploration stage currently prohibits the iterative design of low-level controllers and reward functions. While we currently use grid-based sampling for the task state, other adaptive approaches may scale better for complex tasks. We have investigated a variety of tasks, but we do not investigate tasks which demand precise foot placement.

Chapter 5

Generalized Biped Walking Control

The previous two chapters assumed that low-level SIMBICON controllers are available as input. However, locomotion controllers that are robust to unplanned disturbances and result in a desired motion style and functionality (i.e. walking at a specific speed) are difficult to obtain and require substantial parameter tunning. This chapter introduces a new control strategy designed to address these issues in an integrated fashion and that can therefore be used in a very wide range of scenarios. Specifically, we demonstrate generalization across *gait parameters*, *character proportions*, *motion style*, and *walking skills*.

5.1 Introduction

Physics-based character animation offers the promise of creating motion as it occurs in the world, namely as the product of internal and external forces acting on a skeleton. The nuances that give a motion 'weight', such as foot impacts and subtle balance adjustments, occur as a natural byproduct when working with physical simulations. However, authoring motions in this setting is difficult, given that motions are an indirect end-product of what can be controlled, namely the internal joint torques. Equally important, characters need to be balance aware if they are not to fall over. The development of good balance strategies has proven to be a vexing problem and has been the subject of several decades of ongoing research in computer animation and robotics.

We propose a new control strategy that is inspired by a variety of previous work and consists of four key components. First, a motion generator provides target trajectories that are tracked by proportional-derivative (PD) controllers. Second, a gravity compensation model adds computed torques to the control, which allows for low-gain tracking. Third, balance-aware foot placement is achieved with the help of a predictive inverted pendulum model. Fourth, continuous balance adjustments are produced by using Jacobian transpose control to mimic the effect of a virtual desired force. When further informed by inverse kinematics for the arms and legs, the controller also allows for a variety of flexible interactions with the environment, such as being able to pick

up objects or being able to step at specific locations. These components reinforce one another to achieve a novel control strategy that generalizes in a way that none of the components, taken independently, would achieve.

A wide range of results can be achieved using our method. We demonstrate the integration of walking (forwards and backwards in arbitrary user-controlled directions), standing (quiescent stance idling with foot adjustments), transitions between walking and standing, and multi-step balance recoveries from receiving a push while standing. Our characters can navigate towards a target object placed at any height and reach out to pick it up. Users can interactively create a character of desired proportions and asymmetries and immediately obtain flexible, physically simulated walking. We show that novice users and children can author novel styles with our system. The characters can also push and pull objects, climb stairs, step over obstacles, and lift-and-carry heavy crates.

5.2 Related Work

An expansive body of literature exists on the simulation and control of walking because of its fundamental importance to character animation, humanoid robotics, and human biomechanics. The proposed solutions often aspire to common goals and share a number of common elements.

Foot placement is a key element for achieving robust locomotion, as anyone who has been pushed can attest to. It has been at the heart of many locomotion strategies applied in animation [Raibert and Hodgins, 1991; Laszlo et al., 1996; Hodgins and Pollard, 1997; Yin et al., 2007; Wang et al., 2009] and can be traced back to its origins in robotics, e.g., [Miura and Shimoyama, 1984; Raibert, 1986].

Manipulation of ground reaction forces can play a key role in achieving balanced locomotion. Controllers exploiting this strategy often use full knowledge of the dynamics and current contact configuration. This knowledge allows the most desirable combination of joint torques and ground reaction forces to be computed, which are necessarily coupled and subject to a number of constraints. The desired motion can be specified using a given target trajectory to track or using other locally-defined objectives. This general class of model has been applied to animation [Abe et al., 2007; da Silva et al., 2008a; Muico et al., 2009; Jain et al., 2009; Macchietto et al., 2009] and robotics, e.g., [Takenaka et al., 2009] among many others. Zero-moment point (ZMP) control strategies can also be seen as manipulating ground reaction forces in a strategic fashion.

Learning and optimization hold out the promise of developing and tuning control strategies without necessarily resorting to complex analytical models of the dynamics or careful manual tuning of

parameters [Tedrake et al., 2004; Sharon and van de Panne, 2005; Morimoto et al., 2007; Sok et al., 2007; Wang et al., 2009]. However, it is difficult to develop the right objective functions, and the results are usually specific to a particular character and type of motion.

Simplified models are used in some form in many locomotion controllers [Raibert, 1986; Raibert and Hodgins, 1991; Pratt et al., 2001; Kajita et al., 2003; da Silva et al., 2008a; Yin et al., 2007; Tsai et al., 2009]. Many analytic models of control become inscrutable and intractable when applied to high dimensional systems. Simplified models, such as an inverted pendulum that approximates the motion of the body over the stance foot, provide a low dimensional system that still captures significant aspects of the dynamics.

Characters of varying body proportions can be animated by some kinematic systems, e.g., Spore [Hecker et al., 2008], and has been demonstrated to some extent for a number of dynamic systems [Hodgins and Pollard, 1997; Wang et al., 2009; Tsai et al., 2009; Macchietto et al., 2009]. However, the goal of instantly generating natural dynamic locomotion for multiple skills applicable to arbitrary bipeds remains unsolved.

Our work shares some features of the SIMBICON control method [Yin et al., 2007] and recent related work. These methods have demonstrated the ability to perform variations in style [Yin et al., 2007], to step on or over objects [Yin et al., 2008] and can be optimized to reproduce key features of human walks [Wang et al., 2009, 2010]. The controllers can still be tricky to design, however, because the balance parameters, gait parameters, motion style, and character proportions are not decoupled.

More recently, [Tsai et al., 2009] proposes the use of an inverted pendulum model (IPM) for biped character animation. Importantly, they demonstrate generalization with respect to character proportions, as this is captured by their IPM model. The approach is a *balance filter* method which tracks a motion-capture reference trajectory whose lower-body stepping motion is then modified as dictated by the IPM model.

The Jacobian transpose (JT) application of virtual forces is fundamental to force-based control in robotics. The use of JT methods to generalized control variables, such as the center of mass, was proposed in [Sunada et al., 1994] and further developed in Virtual Model Control [Pratt et al., 2001]. We use JT methods to supply computed-torque gravity compensation, as well as to apply a virtual force to the center of mass which helps regulate its velocity.

As we demonstrate in the results, each of the components of our control strategy (PD control, inverted pendulum, JT gravity compensation, and JT velocity tuning) plays an important role in achieving flexible and agile walking control. Each of these constituent components has existed in at

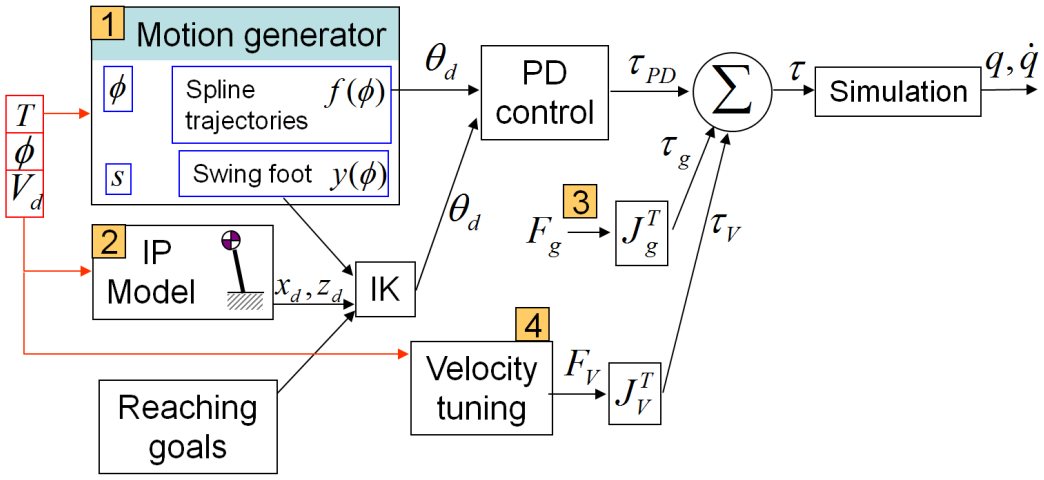


Figure 5.1: System Overview. Key components of the model are: (1) a motion generator for producing desired trajectories; (2) an inverted pendulum model for predictive foot placement; (3) a gravity compensation model for all links; and (4) velocity tuning for fine balance corrections.

least some basic form for 16 years. To the best of our knowledge, however, they have never been integrated to allow for the wide range of generalization, skills, and motion authoring by novice users as demonstrated in this work.

5.3 Control Framework

The control system consists of four key components that are integrated as shown in Figure 5.1. Seen in broad terms, these components work as follows. The motion generator produces open-loop desired joint angle trajectories, which are modeled as spline functions over time. Joint angle trajectories can be modeled relative to their parent-link coordinate frame or relative to the character coordinate frame (CCF), whose y and x-axes are aligned with the world 'up' vector and the character's facing direction, respectively. The use of CCF-relative target joint angles introduces additional feedback into the system because of the implicit knowledge at these joints of which way is up.

A robust balance mechanism is added through the use of foot placement, which is computed with the help of an inverted pendulum model (IPM). The IPM helps achieve motion that is highly robust to disturbances such as pushes. As noted in [Tsai et al., 2009], a feature of IPM-based foot placement is that it does not require parameter tuning because the relevant parameters are captured by the model.

We augment the PD-control with computed-torque gravity compensation. This compensation provides a simple-to-compute estimate of a significant component of the required control, while avoiding the complexity of needing to invert the dynamics of the motion. The addition of virtual gravity compensation forces, F_g, and the torques that implement them, τ_g, allows for acceptable accuracy tracking to be achieved using low-gain PD-tracking in joint space. The low gains, in turn, help allow for natural, highly compliant motion. The virtual gravity compensation forces are computed for each link and are referred back to the root link, namely the pelvis.

We implement fine-scale control by applying a virtual force, F_V, on the center of mass in order to accelerate or decelerate it towards the desired velocity. Like the gravity compensation forces, the torques to implement this are computed using the Jacobian transpose. We now describe each of the components in further detail.

5.3.1 Motion Generator

The *motion generator* produces the various desired trajectories that help create desired motion styles. The trajectories directly model desired joint angles, either relative to their parent joints (elbows, shoulders, stance knee, and toes) or relative to the character coordinate frame (waist, back, neck, and ankles). The character frame is defined by axes aligned with the vertical axis of the world and the facing direction of the character. The desired joint angles are provided as input to PD-controllers that produce tracking torques. The trajectories are modeled as a function of the phase of a step, $\phi \in [0, 1)$ using Catmull-Rom splines. A walk cycle then consists of two steps, with the second step being left-right symmetric to the first. The number of spline segments is arbitrary, although we typically use three segments. The final simulated motions do not necessarily tightly track the target trajectories and contain significant details that are not present in the desired trajectories. The crate lifting and crate pulling are good examples of this (Section 5.4).

The motion generator trajectories can be authored using keyframes, as described in Section 5.5.2. In general, a controller will work robustly with all angle trajectories being set to zero, which implies having the arms pointing straight down, a straight stance leg, feet and toes remaining parallel to the ground, and a fully upright body. This 'zero gait' provides a logical starting point for end-user authoring of gaits. When not zero, the target trajectory for the stance knee is assigned a constant value in order to achieve a bent-leg crouched gait, for example. The stance ankle trajectory can be driven to achieve tip-toe walks or feed-forward toe-off patterns that precede the leg swing. In order to increase the robustness of our characters and the naturalness of the resulting motions, it may be necessary to also add feedback contributions to the ankle trajectories output by the motion generator [Donelan et al., 2009].

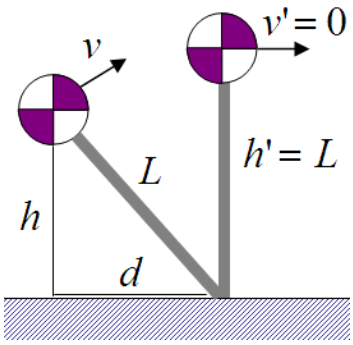


Figure 5.2: Inverted pendulum model.

As will be described shortly, the stance hip and the swing leg are controlled separately and thus these will still actively produce joint motion in the case of all zero joint angle trajectories. Only the $y(\phi)$ curve, which specifies the swing foot height, need be non-zero (Section 3.2).

To allow for simple and coordinated control over bending of the body, the target angles for the lower back, upper back, and head are modeled as fixed ratios of a single user controllable bend angle in the sagittal plane. The constraint can be dropped in order to allow for more artist control and it is not enforced in our style editing interface (Section 5.2).

Given a desired time period for a single step, T, the current phase of an ongoing motion is computed as $\phi = t/T$. The motion generator keeps track of the current stance leg using a two-valued state variable, $s \in \{left, right\}$. A new step is assumed to begin at foot-strike or when $t \geq T$, whichever occurs first. Following [Yin et al., 2007], our motion generator does not provide target trajectories for the stance hip. Instead, its torque is computed to achieve the desired net torque on the pelvis, which is the root link of the character.

5.3.2 Inverted Pendulum Foot Placement

Computing where to step and how to achieve that step is an important problem for walking skills. The steps we wish our controller to perform can vary widely in nature, ranging from making minimal shifts of foot position during idle stance to taking large steps during a fast walk or when receiving a large push.

We compute the desired stepping point, (x_d, z_d), using an inverted pendulum model (IPM) in order to achieve a general solution that works across a wide range of body types. In particular, we build on an inverted pendulum model that assumes constant leg length [Pratt and Tedrake, 2006] as shown in Figure 5.2. The analysis equates the sum of the potential and kinetic energy of the IPM

at the current state, described by its current velocity v, its height, h, and distance, d, from the future point of support, with that at the balanced rest state, i.e., $\frac{1}{2}mv^2 + mgh = \frac{1}{2}mv'^2 + mgh'$, where $v' = 0$ and $h' = L = \sqrt{h^2 + d^2}$. Solving this relation for d gives $d = v\sqrt{h/g}$. The above model computes the desired value of d in order to reach zero velocity at the next step. Taking a shorter step will achieve a positive velocity while taking a larger step will achieve a negative velocity, i.e., walking backwards. Accordingly, we compute $d' = d - \alpha V_d$, where V_d is the magnitude of the desired velocity and α is a constant. We use $\alpha = 0.05$ for all the results we demonstrate. Because the velocity is also controlled using the velocity tuning component, the control is not particularly sensitive to the particular value of α (Section 6). We have experimented with other variations of the IPM, including one that directly takes into account a non-zero desired velocity. Our final choice of IPM as described above is based on the overall robustness and quality of motion. The IPM is applied in the sagittal plane to compute $x_d = d$ and is repeated in the coronal plane to obtain z_d in an analogous fashion. We use the true center of mass position and velocity when applying the pendulum model to the biped. The target values x_d and z_d are updated at every time step, although if a particular foot placement in the world is desired, the character can ignore the IPM prediction for one or two consecutive steps. Depending on the speed of the character, the IPM prediction may be out of reach, in which case the characters uses a maximum step length of $d = 0.6L$. This situation leads to the character possibly having to take multiple steps to recover.

We drive the motion of the swing leg by synthesizing a desired trajectory for the swing ankle relative to the ground and in the character coordinate frame. The height with respect to the ground, y, is modeled as a function of phase by a three-segment spline curve in the motion generator, i.e., $y(\phi)$. This curve defines the difference between a ground-hugging step and a high leg-lift step. The x and z trajectories follow linear trajectories between their locations at the start of the step (x_0, z_0) and their desired location at the end of the step, (x_d, z_d), i.e., $x(\phi) = (1 - \phi)x_0 + \phi x_d$ and $z(\phi) = (1 - \phi)z_0 + \phi z_d$. Inverse kinematics is used to compute target joint angles for the swing hip and knee, which are then tracked using PD controllers and augmented by gravity compensation torques. The inverse kinematics problem has a remaining degree of freedom which allows for knock-kneed, normal, or bow-legged walking variations. We expose this as a twist angle parameter to the animator which is held constant within any given style of motion.

5.3.3 Velocity Tuning

While foot placement ensures robustness for the gait, it can only be enacted once per step. Using foot placement alone ignores the ability to use the stance foot (or feet) to help maintain balance by manipulating the ground reaction forces (GRFs), or equivalently, manipulating the location of the center of pressure (COP), also known as the zero-moment point (ZMP). Simply put, shifting the

COP towards the toes helps in slowing the forward progression of the body, while shifting it back helps accelerate. Many biped walking control algorithms use some form of this manipulation as their primary control strategy. However, this then requires the precomputation of feasible reference trajectories and often assumes full knowledge of the dynamics. In our walking controller, we use a simple virtual velocity-tuning force, akin to that proposed in [Pratt et al., 2001], to provide fine-scale control over the center of mass velocity. It has the effect of altering the GRFs and the COP and so we place this control strategy in the same general category as other techniques that manipulate the GRFs, COP, or ZMP in order to enact balance feedback.

The specific details of computing and applying the virtual force, F_V, on the center of mass are as follows. We first compute the center of mass velocity of the biped, V. In the sagittal plane, we compute a desired virtual force according to $F_V = k_V(V_d - V)$, where V_d is the desired sagittal velocity. In the coronal plane, an analogous virtual force is computed using a PD-controller that tracks a desired lateral COM position. This desired position varies linearly over time from its original position at footstrike to a step width W at the time of the next predicted footstrike. By default, using $W = 0$ results in a catwalk style of motion. A non-zero value of W results in the stance leg pushing the body to the side, which then causes the inverted pendulum model to compensate accordingly with a lateral swing-foot placement.

The combined sagittal-and-lateral virtual force is achieved using $\tau_V = J_V^T F_V$. Here, J_V is the Jacobian of the center of mass for a chain of links that is considered rooted at the stance ankle. In practice, we compute J_V using only the joints between the stance foot and the head, as shown in Figure 5.3 (left), which incorporates all the key joints that support most of the weight. The virtual force helps fine tune both sagittal and lateral balance. Because of the finite support of the feet, applying τ_V can in some situations cause undesired foot rotation, e.g., the toe rotating off the ground in an attempt to accelerate forwards. To mitigate this problem we zero the ankle component of the torque if we detect that the stance foot is not well planted, as measured by the toe leaving the ground or the foot exceeding a rotational velocity threshold. There is potential to have 'chatter' occur, but in practice we have not found this to be the case. During double stance each stance leg is treated independently from its foot to the torso and the results are simply summed.

5.3.4 Gravity Compensation

The use of computed gravity compensation (GC) torques allows for the use of significantly lower gains in the PD controllers for the limbs and the body. Although highly controllable motion could also be achieved with the use of a full inverse dynamics solution, we show that this is unnecessary. The addition of gravity compensation allows simple local control methods, such as low-gain PD-control,

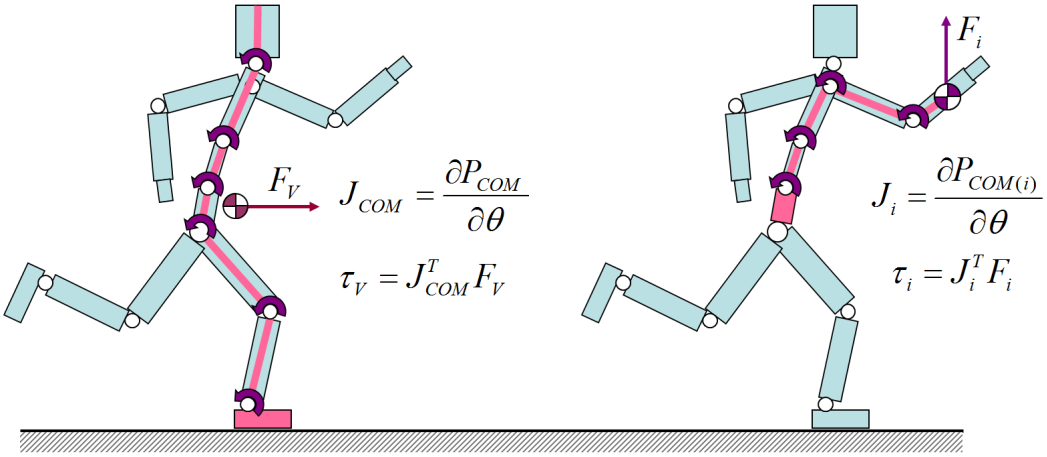


Figure 5.3: Jacobians used for velocity tuning (left) and gravity compensation (right).

to succeed.

Gravity compensation is applied using a Jacobian transpose method once again. We wish to apply a virtual force $F_i = -m_i g$ at the center of mass of every link i, where the negative sign implies an upwards force. The torques required for this are computed as $\tau_i = J_i^T F_i$. Here, J_i is the Jacobian of the link center of mass location with respect to the chain of joints between the root link and link i. GC torques are thus computed for all joints that lie in between each link i and the root link, i.e., the pelvis. Any given joint j thus sees the sum of the GC torques required by all links that are distal to it. The compensation is applied to all links, with the exception of those in the stance leg.

5.3.5 Turning and Limb Guidance

Turning is achieved using the stance hip, similar to [Yin et al., 2007], with a desired turning rate of $\omega_{max} = 2\,\text{rad/s}$. A turning phase, $\phi_{turn} \in [0, 1]$, is defined as the fraction of the turn completed. The desired COM velocity, V_d, is interpolated between its value before the turn and after the turn using ϕ_{turn}. The stance hip twist angle tracks a target facing direction that is interpolated using ϕ_{turn}. The lower-back, upper-back, and head each rotate to anticipate the turn according to linearly weighted functions of ϕ_{turn} so that the head leads the anticipatory movement. The swing-leg IK plane of rotation also anticipates the turn, which helps plant the swing foot in an anticipatory fashion.

During sharp turns, the default trajectory for the swing leg may cause it to collide with (or in

our case, pass through) the stance leg. If the swing foot trajectory passes on the wrong side of the stance foot, a 3-segment spline curve is computed based upon the current position, the target position, and a fixed waypoint. This strategy greatly reduces the number of self collisions, although in our implementation it does not completely eliminate them. Analysis of human swing leg motions is a possible principled alternative for dealing with this issue, but it is unclear if it would generalize well across gaits, styles, and character proportions.

Cartesian-space trajectories are used to guide the motion of the swing foot, as presented earlier. Given the current world-coordinates target position for the swing foot, inverse kinematics is used to compute the target angles for the swing hip and knee. With this approach, a bent stance knee automatically results in a matching bent swing leg motion since the hips are lower. This approach also allows the controller to guide the swing foot over an obstacle by only changing the desired foot height. If needed, the foot target location computed by the inverted pendulum model can be ignored in favor of precisely positioning the foot at a specific ground location. However, this typically cannot be done for multiple steps in a row since it ignores balance considerations.

Physics-based characters need to be equipped with skills that go beyond basic walking if they are to be used extensively in games and simulations. One fundamental skill is that of being able to reach towards specific points in the environment in order to pickup an object, place a hand on a door handle, or to turn on a light switch. In our framework, the desired joint angles for the shoulder and elbow are computed with inverse kinematics once the character is within reaching distance. Gravity compensation is key to achieving successful motions without resorting to excessively high PD gains.

We employ an analytic solution to the two-link inverse kinematic problems [Kulpa et al., 2005]. To compute a unique solution, the elbow or knee is forced to lie in a plane that also embeds the shoulder and hand, or analogously for the leg, the hip and ankle. The rotational degree of freedom of this embedding plane is specified as an input parameter that allows for bow-legged walks or control over the style of arm reaching movements.

5.4 Implementation

A version of our implementation will be made available as an open source project. We use *Open Dynamics Engine* [ODE, 2010] as our forward dynamics simulator. A simulation time step of 0.0005 s is used. All balance control parameters are kept fixed across all our simulations, i.e., there are no parameters to tune by a user interacting with the system. For PD-gains, we use k_p values that scale with the total mass of the character being controlled. Specifically, when an edit creates a new character with mass M', we scale the k_p values by M'/M, where M is the original character

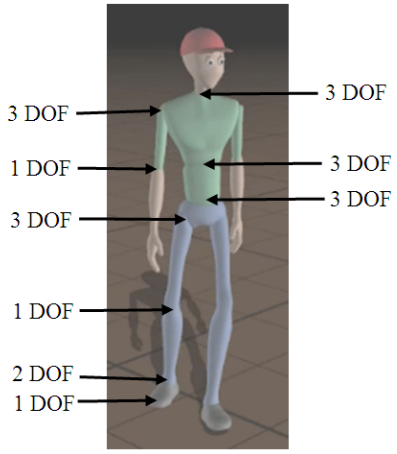


Figure 5.4: Character degrees of freedom.

mass. We use $k_d = 2 * \sqrt{k_p}$ for all joints. Without PD scaling, the first point of failure is typically for large characters whose knees begin to buckle under their weight. The coefficient of friction between the character and the ground is 0.8. We do not use torque limits in the specific results we show. However, we have experimented with adding torque limits ($|\tau| \leq 200Nm$, for hips and knees, $|\tau| \leq 100Nm$ for all other joints) for our humanoid character and this did not have any visible impact on the motions. With the exception of a simulated crowd scene, all the simulations run in real time, written as a single-threaded application and run on a 2.4GHz 2-core machine with 2Gb of memory. We do not process collisions between body parts belonging to the same character.

Our characters have the degrees of freedom shown in Figure 5.4. There are a total of 37 DOF, which includes 6 DOF for the global position and orientation. Our *humanoid* character (Figure 5.5, row 1) has dimensions and a mass distribution that are typical of a human. The *robot* character (Figure 5.5, row 4) has similar dimensions, though its left arm is 40% longer and heavier, and the left knee is 20 cm higher than the right knee. The *beast1* and *beast2* characters (Figure 5.5, rows 2 and 3) have much of their mass in the link that corresponds to their pelvis. Characters created using the character editor have link masses that are proportional to their volume, i.e., the density is assumed to be constant.

5.5 Results

We demonstrate a variety of animation results.

5.5.1 Generalization Across Gait Parameters

We first show that the walking control generalizes across forwards-and-backwards walking, walking speeds, stepping frequencies, and turning towards a desired direction. We also show integrated stopping and starting behaviors. We demonstrate these basic behaviors across different characters and for different styles of motion. Figure 5.5 shows frames from the animations. A commanded forward walk is indicated by a blue arrow, a backwards walk by a red arrow, and a green dot indicates a command to stop walking.

The desired speed, V_d, is given as an input to the control system, as shown in Figure 5.1, and is achieved using a combination of foot placement (via the inverted pendulum model) and Jacobian transpose control (velocity tuning). The desired time period, T, is an input to the motion generator, where it is used in computing the current phase, $\phi = t/T$. Walking with time periods of $0.2\,\mathrm{s} \leq T \leq 1.0\,\mathrm{s}$ works well for characters of approximately human scale. Very long time periods ($T > 1s$) with medium-to-large steps are problematic as the character needs to pause when over the stance foot, while our control strategy assumes a constant desired speed. The average stepping length, L, is implicitly determined by $L = V_d T$. Setting $V_d < 0$ results in backwards walking. Our characters walk at speeds ranging from -0.6 to $1.7m/s$.

When the character is stopped, balance requires no special treatment – it is achieved using the velocity tuning component, with the torques being referred to both the right and left feet. The sagittal and lateral components of the virtual force F_V are both computed using a PD-controller that now tracks a static target position for the COM that is located at the midpoint between the two feet. To begin walking, the COM-target is simply switched to lie in the middle of the desired future stance foot. Once the COM has moved 40% of the way, walking is triggered. This trigger event is equivalent to a footstrike in the walking model, i.e., it represents the initiation of a new step. To stop walking, V_d is set to zero. Once $V \leq \epsilon$ is achieved at footstrike, which may require more than one step, the standing balance control is invoked as described earlier. An idling behavior is used to create quiescent stance motions that shift the weight between the legs and to make small adjustments of the unweighted foot. The behavior is scripted using a simple finite state machine.

The character can also recover from pushes while standing by invoking walking when needed, using $V_d = 0$ and the same stop-walking condition described above. The COM position and velocity is monitored while standing. If the COM position is too close to the edge of the support polygon or $V > \epsilon$, then walking is invoked in order to regain balance by stepping.

We demonstrate control over gait parameters for characters of varying proportions and for different styles in order to show generalization across the cross-product of these features. Figure 5.5 illustrate

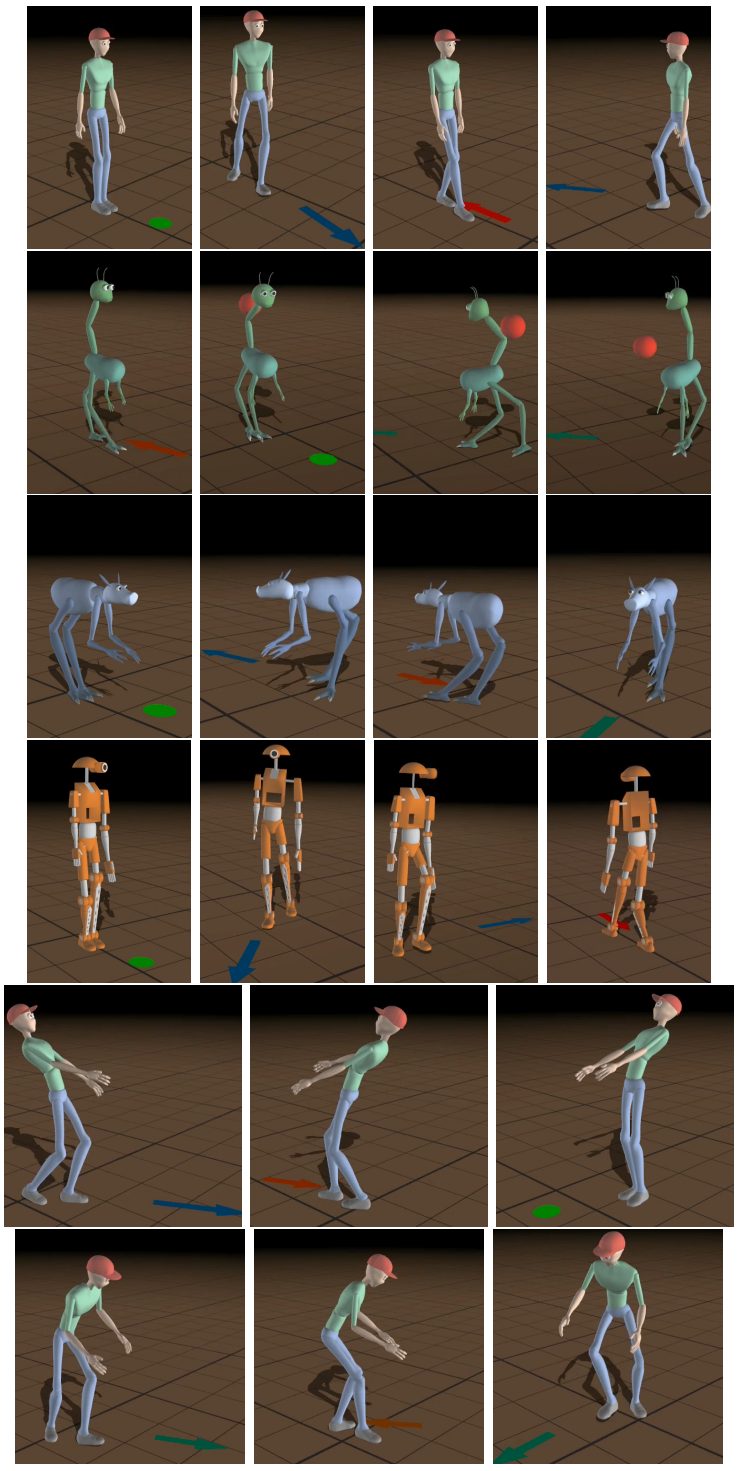


Figure 5.5: Direction following.

the execution of a sequence of commanded gait parameters over four characters of varying proportions and dimensions. No character-specific or style-specific parameter tuning is required.

5.5.2 Generalization Across Style

The controller produces balanced motions for a wide variety of user-authored styles of motions. Rows 1, 5, and 6 in Figure 5.5 illustrate three different walking styles for the same character. These styles were authored by modifying the lean of the upper body in the sagittal plane, the target angle of the stance knee, the plane of rotation for the swing-leg IK, and the arm motions.

We also develop an interface that allows animators to easily author unique motion styles. Figure 5.6 shows the interface and example motion styles that were created in 5-10 minutes by novice users, including children as young as 7. Our walking control strategy instantly provides robust dynamically-simulated gaits. The interface is constrained to symmetric trajectories for the sake of simplicity.

5.5.3 Generalization Across Characters

The control strategy generalizes well across a variety of character dimensions and proportions, as shown in rows 1–4 of Figure 5.5. We further develop an interface that allows a user to interactively edit the dimensions and proportions of a character and immediately see the resulting motions. The interface and examples of resulting characters are shown in Figure 5.7. Characters can have asymmetric arms and legs, which will introduce asymmetries in the motion even with symmetric controls from the motion generator. Asymmetry in the placement of the knee joint, such as for our robot character, is largely accommodated by the inverse kinematics used for tracking the desired swing ankle trajectories.

5.5.4 Generalization Across Tasks

Walking controllers should also generalize across tasks and be able to interact with the environment in various ways. We demonstrate this on several different tasks.

Reaching: Figure 5.8 shows the ability to reach and grasp an object placed at an arbitrary location and at an arbitrary height. We demonstrate object reaching motions on three different characters, showing generalization across character proportions. The reaching strategy uses a high-level behavior that controls the walking direction in order to guide the character towards the target object. A desired heading direction is computed once per step for the character based on

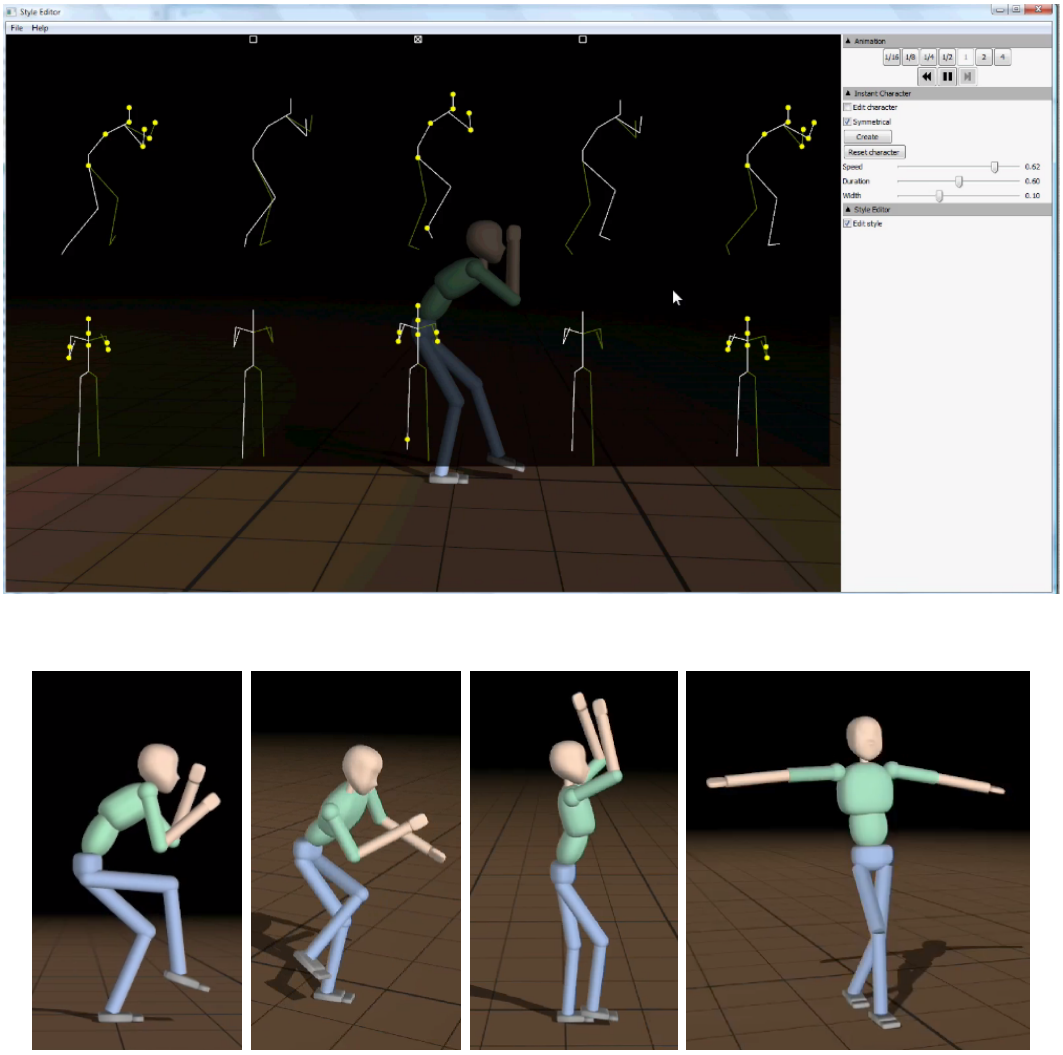


Figure 5.6: Motion style editing and example motion styles.

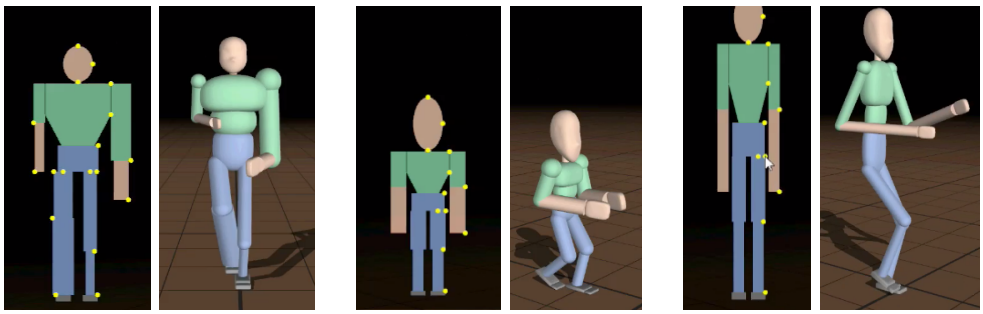


Figure 5.7: Interactive editing of character proportions and the resulting walking gait.

the location of the object. Most reaching motions can be carried out while walking at a regular pace. Objects near the ground require the character to slow down, which is implemented as a linear function of the remaining distance, d, to the object. Low reaches also require bending down, using a combination of forward upper body lean and bending of the stance knee. The reaching motion is triggered when $d < \epsilon$, at which point the arm treats the final object position as a target. Inverse kinematics provides required joint angles, which are then tracked using low-gain PD-controllers and augmented by the gravity-compensation torques.

Pulling and pushing a crate: The walking control can be used to pull and push objects, as shown in Figure 5.9. The crate weighs 80 kg and has a coefficient of friction of 0.2 with the ground. The high level behavior has the character walk towards the crate, slow down when near the crate handles, and reach for the handles when within reach. Springs and dampers are used to connect the hands to the handles. The hands are not animated in our implementation and are treated as an extension of the lower arm. The user can interactively control the target bend parameter of the body to produce sporadic tugging behavior, if desired. The inverted pendulum model is adapted to account for the weight to be pushed or pulled by using $\alpha = 0.2$ instead of $\alpha = 0.05$.

Lifting and moving a crate: The simulated characters can lift and move heavy crates (5–25 kg) using the generalized walking control, as shown in Figure 5.10. The high level behavior used to approach the crate is similar to that used in the pulling and pushing skill. Once the hands are attached to the box with stiff springs and dampers, the crate mass is incorporated into the computations for the overall COM position and velocity. It is also incorporated into the gravity compensation by adding a compensation force corresponding to half the weight of the box to each hand. The hands are not animated. The user controls the walking speed and direction interactively once the character has lifted the crate.

Navigation over and under obstacles: A high level behavior that steps over obstacles and ducks under other obstacles is also easy to create. Of particular interest is that the stepping-over

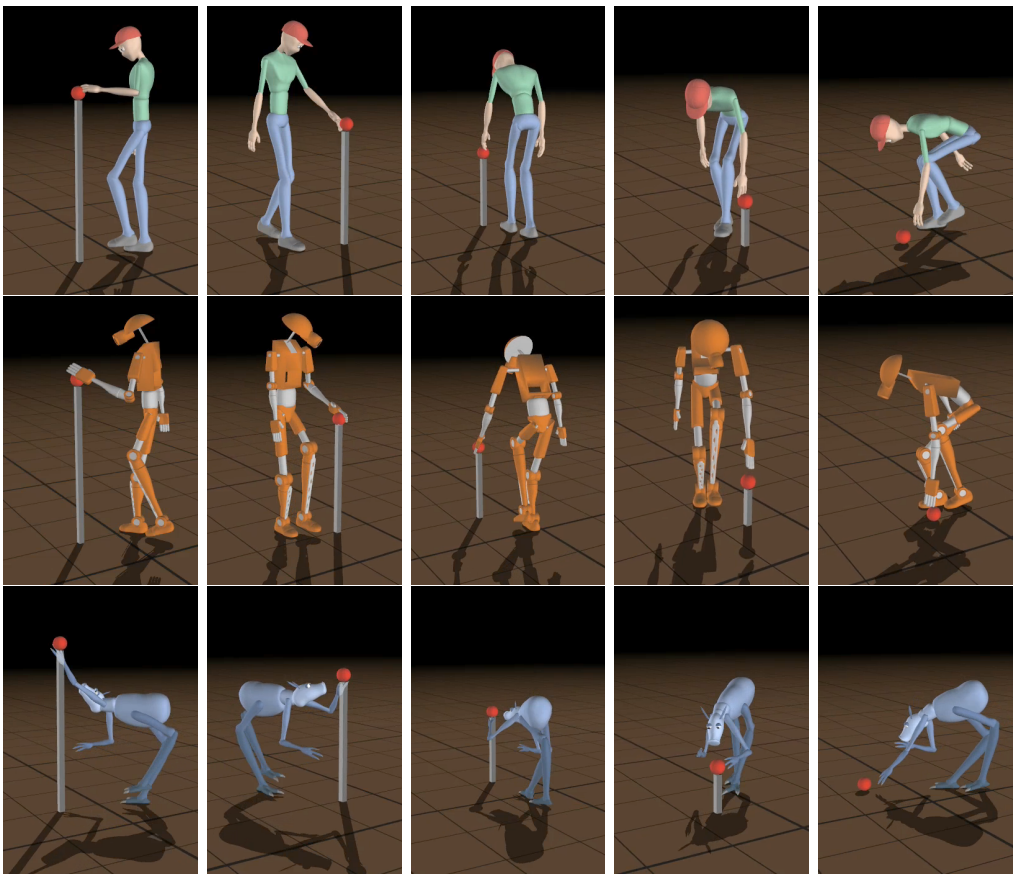


Figure 5.8: Reaching for objects

behavior and the ducking behavior are controlled independently as they do not interfere with each other. Figure 5.11 shows results for two different humanoid characters in navigating an obstacle course. The step-over and duck-under obstacles are paired together with varying offsets in order to demonstrate that the factored control works well for all combinations of these two skills. The control for ducking under obstacles consists of bending the body as a linear function of the obstacle distance, yielding a smooth ducking motion.

The high level control for stepping over an obstacle involves slowing down upon approaching the obstacle. It is then important to arrange for a footstep that is planted near the obstacle itself before stepping over it. This footstep is done by adapting the step length once within a threshold distance of the obstacle. Because the location of the swing foot is determined by the inverted pendulum model, we implicitly adjust the step length by manipulating the desired velocity. Once in position to step over the obstacle, a trajectory is planned that allows the swing foot to clear the obstacle in the upcoming step. This trajectory guides the swing ankle and is constructed using 4 Catmull-Rom Spline segments. The swing foot follows this trajectory until the heel is past the obstacle, after

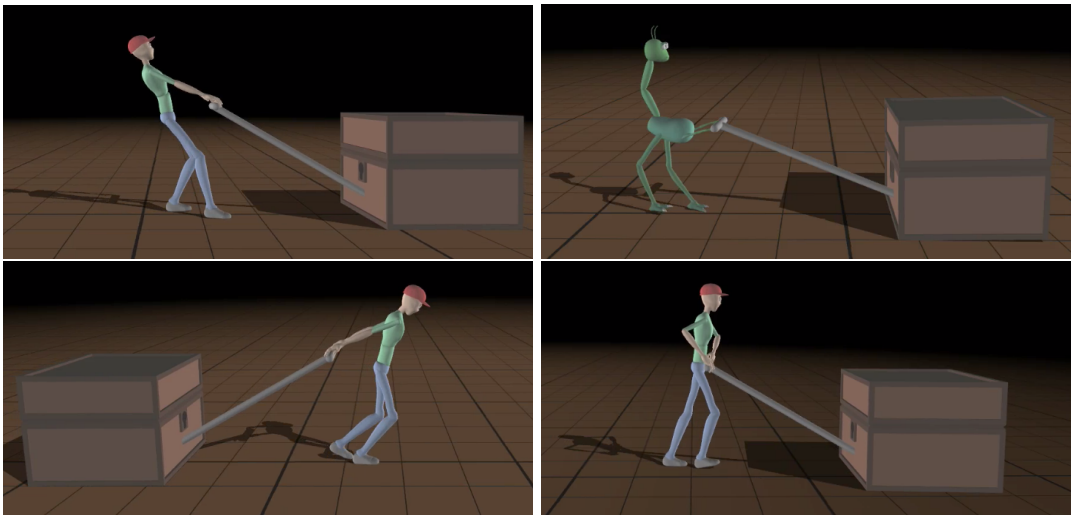


Figure 5.9: Pulling and pushing a crate. Top: Pulling to the left. Bottom: Pulling and pushing to the right.

which control of the forward component once again reverts to the inverted pendulum in order to produce a final foot placement that is suitable from the point of view of balance. The motion of the trailing foot over the obstacle is controlled in an analogous fashion.

Stairs: Figure 5.12 shows the humanoid character carrying a 15 kg crate as it walks up 15 cm stairs and steps over 20 cm obstacles. The high level control for climbing a step is identical to that for stepping over an obstacle, except for the way in which the vertical trajectory of the swing ankle is computed.

Crowd Simulation We simulate a crowd of 16 dancing characters of various sizes, shown in Figure 5.13. The characters occasionally bump into each other or step on each other's feet, which creates a scene that is rich in motion detail. Creating this scene simply requires instancing 16 characters with randomized proportions and gait parameters and no other special effort. The simulation runs approximately 10-times slower than real time. The simulation speed varies as the number of contacts increases and decreases.

5.6 Discussion

The use of the suitable control representations and simplified models allows for significant generalization to occur across body types and motion styles without needing to resort to learning or needing to invert the full dynamics.

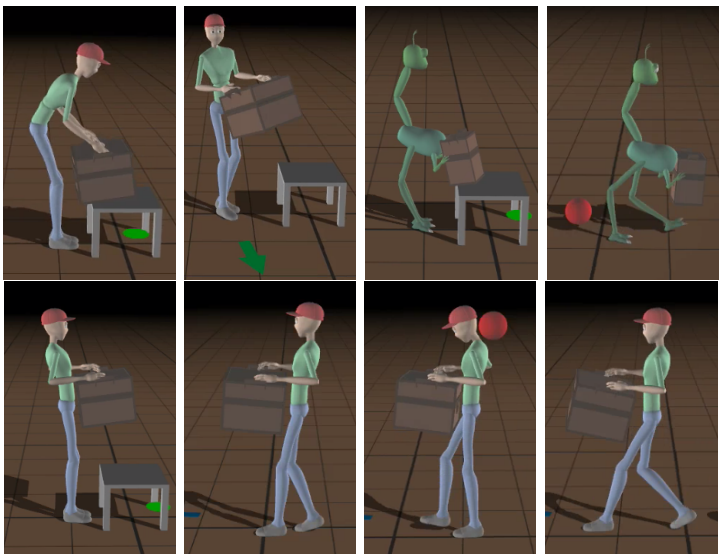


Figure 5.10: Lifting and moving heavy crates.

Representations and generalization: An important test of the utility of a motion control model is to see how well it generalizes across a range of motions and characters. In our results, we demonstrate a significant degree of generalization across gaits, styles, characters, and skills. The control strategy achieves much of its power from the way the control is factored into several components which can then be composed together. The individual components each see only a limited aspect of the overall dynamics.

A viable alternative approach is to assume that the dynamics are fully known and to exploit this in the control strategy [da Silva et al., 2008a; Macchietto et al., 2009; Jain et al., 2009]. In particular, a model of the inverse dynamics allows a target trajectory to be tracked, and in this way motions can be represented directly by desired motion trajectories. The presence of underactuation and contacts make this a challenging problem for walking, but with appropriate prediction these can also be taken into account [Muico et al., 2009]. One way to achieve generalized control within a trajectory-tracking model might be to parameterize the input physically-feasible target motion trajectories with respect to gait, style, character dimensions, and skills. However, this is a challenging problem to solve and it resembles the desired output of an animation system rather than an input.

Our proposed control strategy is constructed around approximate models that can be used in support of executing new motions with a reasonable degree of skill. It lends some support to the hypothesis that, with the right representation, *"Bipedal walking is an inherently robust problem, not requiring precision, accuracy, or repeatability"* [Pratt and Drakunov, 2007]. The computational requirements of the control are low as no implicit or explicit inversion of the system dynamics is required. The method does not need online or offline optimizations.

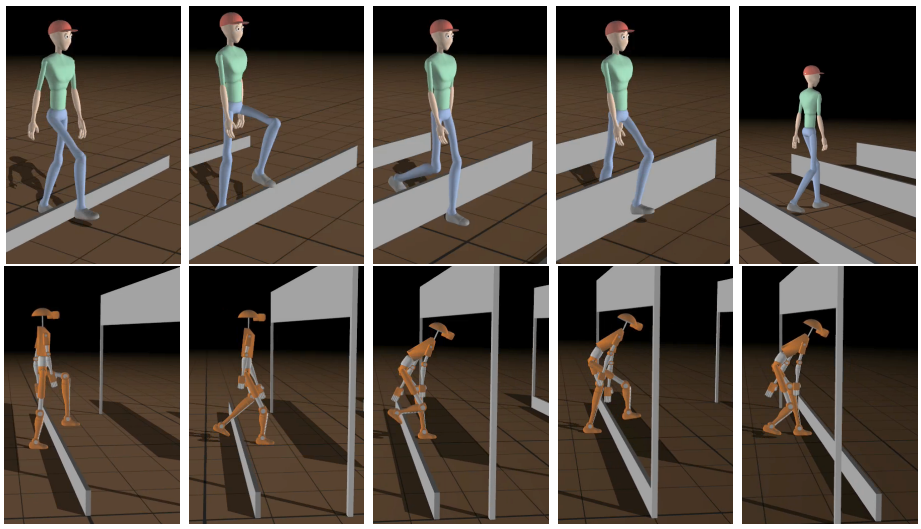


Figure 5.11: Navigating over and under obstacles. Top: Stepping over a sequence of obstacles of up to $45cm$ in height. Bottom: Stepping over a sequence of obstacles combined with ducking under obstacles, with varying offsets.

Figure 5.12: Walking to a crate, picking it up, climbing steps, stepping over obstacles, and coming to a stop.

Anticipation: Skilled motion generally requires a strong degree of anticipation of the forces required to perform a motion. There are several anticipatory elements in our control strategy. The inverted pendulum is a simplified dynamics model that makes specific predictions regarding foot placement to achieve a desired velocity. The gravity compensation model anticipates a significant degree of the control required for low-speed motions. The PD-controllers only need to deal with inertial dynamics. The velocity tuning has knowledge of the mass distribution of the body and its kinematics in relation to the point of support. Taken in combination, these control elements exhibit a significant degree of anticipation. To achieve further anticipatory control, we expect that the PD control could be largely replaced by feed-forward torques that are learned [Yin et al., 2007] or that are computed from a reference trajectory using inverse dynamics.

Authoring: Computer animation tools need to provide artist-friendly representations. We have

Figure 5.13: A dancing crowd with 16 characters.

developed a simple interface to support interactive editing of motion styles and character proportions. Additional parameters, such as the turning rate can also be exposed as high level parameters. With these tools, users with no prior animation experience are able to create highly varied characters and motion styles in 5–10 minutes during unconstrained exploration, i.e., we do not ask them to reproduce a specific reference motion style. The rules that encode high level skills such as "go pick up the crate and carry it to here" are currently scripted.

Robustness: We test the robustness of the humanoid character by applying external disturbance forces to the torso for a duration of 0.1 s. During a forward walk of 0.6 m/s, the character is robust to omni directional pushes of 600 N, applied at arbitrary points in the walk cycle. The mass of the character is $70.4kg$.

Parameter settings: We use fixed parameter settings across all motions, unless otherwise noted. These include $\alpha = 0.05$, the inverted pendulum velocity gain; $k_v = 100Ns/m$, the velocity tuning gain; and $\omega_{max} = 2$ rad/s, the maximum turning rate. A default trajectory needs to be provided for the height of the swing ankle as a function of phase, and a self-collision avoidance strategy needs to be designed that helps avoid the swing leg colliding with the stance leg during sharp turns.

We investigate the effect of doubling and halving the PD-gains used for joint tracking. The overall motion remains similar, although high-gain tracking results in a stiffer-looking motion, while the very low-gain tracking gives a slightly sloppy and weak looking motion. We have also examined the effect of changing α, the parameter that modifies the prediction of the inverted pendulum mode. A value of 0 works well for desired speeds $|V_D| \leq 0.4$. For larger V_D values, the walk cycle can become asymmetric, as more energy is lost due to the location of the step than can be injected during one step by the virtual model control. We have also tried doubling the value of α, without any noticeable decrease in robustness. However, as the value of α keeps increasing, the predicted foot location may become unsuitable for maintaining balance.

Contributions of each control component: Each of the core components plays an important role in achieving robust and flexible control. The effect of removing gravity compensation results in a slouched upper body, unsuccessful reaching motions, and less accurate placement of the swing foot. Removing the inverted pendulum control results in an inability to recover from any significant perturbations. Without velocity tuning the velocity is controlled with much less precision and the characters have difficulty walking at steady state for a given velocity.

Strengths and weaknesses: Section 2.3 presents a list of desirable attributes that can be used to compare the relative strengths and weaknesses of existing control methods. The control strategy introduced in this chapter does particularly well in terms of the following criteria:

- Robustness to unplanned perturbations: The combination of foot placement control via the inverted pendulum model, and JT control used for fine-level velocity tunning allows our characters to recover from large external perturbations and quickly return to a steady-state walk.
- Diversity of motions: A variety of motions were produced with our control strategy, ranging from simple walking in a straight line, to stepping over and ducking under obstacles and pushing, pulling and carrying heavy crates.
- Generalization to different body proportions: The control strategy we presented generalizes to bipedal characters of a wide variety of body proportions without the need for any manual parameter tuning.
- Robustness with respect to control parameters: High-level gait parameters, such as the step height, stride duration, desired velocity and heading can be manually edited, without requiring any other parameter tunning. A significant range of values can be used for the PD gains as well.
- Ease of implementation: Only PD controllers and cross products for the Jacobian computation are needed to implement the control strategy.

Our control framework also has a number of limitations. While the current framework allows

authoring a motion style, it does not allow for the authoring of a 'push recovery style', as this aspect of the motion is currently governed by the inverted pendulum foot-placement. Self-intersection between the swing and stance legs still occurs in some situations. We do not demonstrate generalization across terrain, with the exception of climbing steps. We have not explored extending the method to non-biped morphologies. The motions that we produce are agile in the sense of being able to achieve fast turning behavior and fast transitioning between forwards and backwards walking. However, we have not demonstrated high-speed agility. Motions which are particularly demanding with respect to energy or other performance measures may still be challenging to author and would benefit from optimization.

Chapter 6

Conclusion

6.1 Discussion

Simulated models of motor control are applicable to a wide range of applications. They can be used, for instance, to synthesize the motions of characters in interactive applications, control legged robots as they locomote in urban environments and drive neuroprosthetic devices that may some day improve the lives of people with physical disabilities.

This thesis outlined several approaches to constructing control policies that allow physically-simulated characters to demonstrate skill and purpose as they accomplish higher-level locomotion tasks. In Chapter 3 we showed how a parameterized walking controller can be used, in conjunction with a finite-horizon planner, to allow characters to navigate terrains with obstacles that need to be stepped over. In Chapter 4 we examined a different problem related to locomotion. Given a set of individual walking skills such as walking, turning and running, we presented a way of automatically pre-computing control policies that allow a character to quickly move to a given location or follow a user-directed heading. The specific details for these two frameworks that couple high-level planning with low-level control are summarized in Table 6.1.

	Constrained Walking (Chapter 3)	**Task-level Control** (Chapter 4)
Input	One SIMBICON controller, sample "stepping stone" problems	Set of SIMBICON controllers controllers
Action space	continuous	discrete
Planning approach	finite horizon (1 or 2-step look ahead), online	infinite horizon, pre-computed
Planning frequency	once per step	once per step
Task	Terrain navigation	Higher-level locomotion goals

Table 6.1: Overview of planning methods

The two proposed methods have their own strengths and weaknesses. Consider, for instance the task

of getting to a target location as quickly as possible. The finite-horizon planner used for Constrained Walking (CW), would be unable to determine when, or if, the character should start running by looking only two steps into the future. Increasing the planning horizon may help, but finding an agile solution would involve predicting the results of an exponential number of possible actions, which is too computationally expensive for a real-time planner. In addition, the predictive power of our system diminishes as the planning horizon is increased due to accumulating errors. The infinite horizon planning strategy that we use for Task-Level Control (TLC), in contrast, is able to learn under what circumstances transitioning to a run is beneficial in the long term, and when the character should start to slow down so that it does not overshoot its goal.

The TLC framework, however, requires a parameterization of the task that the character is trying to achieve. Parameterizing rough terrain is not straightforward because it is unclear how far ahead the character should be looking. Our experiments further indicate that pre-computing control policies for tasks whose parameterizations are more than three-dimensional becomes prohibitively expensive. The CW framework, to a large extent, bypasses this problem. As long as a mechanism is in place that allows the planner to predict the results of available actions, and a short-planning horizon suffices, this approach will be successful. Another important feature of CW is that it automatically builds its own continuous action space while discovering how to solve example problems from the task domain. This approach is particularly interesting because, to some extent, it mimics the processes used by humans and animals as they learn and fine-tune various motor skills.

In Chapter 5 we introduced Generalized Biped Walking Control (GBWC), a low-level control strategy meant as a replacement for SIMBICON, which served as a starting point for CW and TLC. In contrast to SIMBICON, GBWC offers precise control over high-level gait parameters such as walking speed, stride frequency, swing foot trajectory and heading without the need to tune any other parameters. By manipulating these high-level gait parameters, we were able to achieve a variety of interesting behaviours such as pulling, pushing, lifting and carrying heavy crates, picking up objects from varying heights, turning, walking at various speeds and navigation of terrains with obstacles that need to be stepped over, under and onto. With this new control model we were able to re-create some of the behaviours that had been demonstrated through our higher level planners, but without the added complexity.

6.2 Conclusion

Developing locomotion skills for simulated characters is a challenging problem, given the high-dimensional, non-linear nature of the input and output spaces the control policies have to deal with, as well as the inherently unstable nature of bipedal creatures. In this thesis we showed that through

the use of appropriate levels of abstraction, developing high-level locomotion skills that go beyond simple walking in a straight line becomes a tractable problem.

The use of balance-aware locomotion controllers as low-level actions by our higher level planners presents important benefits. The locomotion controllers continuously drive the motion of the characters for relatively long periods of time and they mimic reflexes by providing an appropriate first response to unplanned disturbances. This allows our planners to be invoked on a step-by-step basis, which reduces their complexity.

In order to map the result of the low-level actions across the high-dimensional character state-space, so as to allow the planners to make appropriate decisions, we used a discrete dynamics model. The dynamics model is built over subsets of the state space that are likely to be reached given the available set of actions. One important observation is that the more reliable the low-level actions are, the easier the planning process becomes. For this reason we introduced a new low-level control strategy that generalizes well across gait parameters, motion styles, character proportions, and locomotion tasks. The method works by abstracting away the character and dealing mainly with its center of mass and swing foot trajectory. In the future, it is our goal to explore the use of this new control strategy in conjunction with higher-level planners.

6.3 Future Work

The work presented in this thesis opens the door to many promising directions for future work. Our simulated characters do not yet demonstrate the agility that humans are capable of. We would like to see, for instance, virtual characters that are capable of playing tennis. One possible approach to achieving this would be to emulate the learning processes that humans go through as they learn skills. Continuous action spaces can be automatically discovered by having our characters learn how to perform increasingly more agile motions, in a process similar to the one used in Constrained Walking. Example data in the form of motion capture or force plate information can also be used to steer the learning process towards human-like motions.

At the other end of the spectrum lies a different category of motions - motions representing small, purposeful foot-placement adjustments that are often performed in order to better position oneself for the task at hand. Sitting on a chair, opening a door, or washing dishes are just a few examples of tasks where these subtle motions can be very important. Observing the strategies that people use in day-to-day motions can again serve as a great starting point. Certain features of the motion, such as the trajectory of the center of mass relative to the support polygon, could provide meaningful, low-dimensional information related to the control principles used by humans. This information

can be treated as observations of expert control policies for the problem of Inverse Reinforcement Learning (IRL) [Ramachandran and Amir, 2007; Russell, 1998; Ng and Russell, 2000; Atkeson and Schaal, 1997]. The example motions can also be used to determine the regions of the state spaces that the policy building process should focus on.

People and animals, both real and simulated, unavoidably fall at times - admittedly, simulated characters fall a lot more. Motor control strategies aimed at helping them get back up are vital. However, the modeling of such motor control skills has not received much attention. Faloutsos et al. [2001] presented a system where rising motions are generated, mostly thanks to hand-tuned control strategies. To date it has not been shown that these control strategies are general-purpose enough to be used in every-day scenarios. An interesting direction for future research would involve examining and reproducing strategies that people use when they have to get up from sitting or lying positions. Of particular interest would be the exhibited pattern of foot and hand contacts, especially for interesting scenarios where helpful props are available. Inverse Reinforcement Learning ideas may be useful for this endeavor as well.

Control of bipedal locomotion skills has received a great deal of attention in recent years. Models of motor control, however, hold the promise of being able to plausibly driving the motion of creatures with arbitrary number of legs. Simulating the various gait transitions that a horse goes through as it slowly begins to gallop, for instance, would be an interesting challenge. Once control strategies are validated on data recorded from real animals, this methodology can potentially be used to synthesize the motions of extinct or imaginary creatures.

Another interesting direction to pursue is the integration of our control methods with accurate biomechanical models. Incorporating realistic delays, muscle actuators that span multiple joints and accurately modeled joints may increase the naturalness of the resulting motions. Implementing our work on real-life legged robots is also a very exciting direction for future work. We speculate that the Generalized Biped Walking Control framework is particularly well suited for this task, as it does not require a dynamics model and it only uses information that is readily available to be measured or estimated.

Bibliography

Abe, Y., da Silva, M., and Popović, J. (2007). Multiobjective control with frictional contacts. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 249 – 258.

Alexander, R. M. (1995). Simple models of human motion. *Applied Mechanics Review*, 48(8):307 – 316.

Alexander, R. M. (2003). *Principles of Animal Locomotion*. Princeton University Press.

Arikan, O. and Forsyth, D. A. (2002). Interactive motion generation from examples. *ACM Transactions on Graphics (SIGGRAPH)*, 21(3):483 – 490.

Atkeson, C. G. and Schaal, S. (1997). Robot learning from demonstration. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 12–20.

Atkeson, C. G. and Stephens, B. (2007). Random sampling of states in dynamic programming. *Proc. Neural Information Processing Systems Conf.*, 38(4):924 – 929.

Beaudoin, P., van de Panne, M., Poulin, P., and Coros, S. (2007). Motion-motif graphs. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 117–126.

BostonDynamics (2010). Littledog - the legged locomotion leanring robot, `http://www.bostondynamics.com/robot_littledog.html`.

Boulic, R., Mas, R., and Thalmann, D. (1996). A robust approach for the control of the center of mass with inverse kinetics. *Computers and Graphics*, 20:693–701.

Brand, M. and Hertzmann, A. (2000). Style machines. *SIGGRAPH '00 Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183 – 192.

Byl, K. and Tedrake, R. (2008). Approximate optimal control of the compass gait on rough terrain. In *International Conference on Robotics and Automation (ICRA)*, pages 1258 – 1263.

Chai, J. and Hodgins, J. K. (2007). Constraint-based motion optimization using a statistical

dynamic model. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):Article 8.

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

Chestnutt, J. and Kuffner, J. (2004). A tiered planning strategy for biped navigation. In *Proceedings of the IEEE - RAS / RSJ Conference on Humanoid Robots*, pages 422 – 436.

Chestnutt, J., Lau, M., Cheung, K. M., Kuffner, J., Hodgins, J. K., and Kanade, T. (2005). Footstep planning for the honda asimo humanoid. In *International Conference on Robotics and Automation (ICRA)*, pages 629 – 634.

Choi, M., Lee, J., and Shin, S. (2003). Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203.

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

da Silva, M., Abe, Y., and Popović, J. (2008a). Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):Article 82.

da Silva, M., Abe, Y., and Popović, J. (2008b). Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum*, 27(2):371 – 380.

da Silva, M., Durand, F., and Popovic, J. (2009). Linear Bellman combination for control of character animation. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):Article 82.

de Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 131.

Donelan, J. M., McVea, D. A., and Pearson, K. G. (2009). Force regulation of ankle extensor muscle activity in freely walking cats. *Journal of Neurophysiology*, 101(1):360–371.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. (2001). Composable controllers for physics-based character animation. *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 251 – 260.

Heck, R. and Gleicher, M. (2007). Parametric motion graphs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 129–136.

Hecker, C., Raabe, B., Enslow, R. W., DeWeese, J., Maynard, J., and van Prooijen, K. (2008). Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):Article 27.

Hobbelen, D. G. E., de Boer, T., and Wisse, M. (2008). System overview of bipedal robots flame and tulip: Tailor-made for limit cycle walking. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2486–2491.

Hodgins, J. K. and Pollard, N. S. (1997). Adapting simulated behaviors for new characters. *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 153–162.

Hodgins, J. K. and Raibert, M. N. (1991). Adjusting step length for rough terrain locomotion. *International Conference on Robotics and Automation (ICRA)*, 7(3):289 – 298.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71 – 78.

Hofmann, A. G. (2006). *Robust Execution of Bipedal Walking Tasks from Biomechanical Principles*. PhD thesis, Massachusetts Institute of Technology.

Huang, P. S. and van de Panne, M. (1996). A planning algorithm for dynamic motions. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, pages 169–182.

Ikemoto, L., Arikan, O., and Forsyth, D. A. (2005). Learning to move autonomously in a hostile world. Technical Report UCB/CSD-05-1395, EECS Department, University of California, Berkeley.

Jain, S., Ye, Y., and Liu, C. K. (2009). Optimization-based interactive motion synthesis. *ACM Transactions on Graphics*, 28(1):Article 10.

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 (ICRA)*, pages 1620 – 1626.

Khatib, O., Sentis, L., Park, J., and Warren, J. (2004). Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43.

Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics (SIGGRAPH)*, 21(3):473 – 482.

Kuffner, J., Nishiwaki, K., Kagami, S., Kuniyoshi, Y., and Inoue, H. (2003). Online footstep planning for humanoid robots. In *International Conference on Robotics and Automation (ICRA)*, pages 932–937.

Kulpa, R., Multon, F., and Arnaldi, B. (2005). Morphology-independent representation of motions for interactive human-like animation. In *Computer Graphics Forum*, volume 24, pages 343–352.

Kuo, A. D. (1999). Stabilization of lateral motion in passive dynamic walking. *The International Journal of Robotics Research*, 18(9):917–930.

Kwon, T. and Hodgins, J. (2010). Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Kwon, T. and Shin, S. Y. (2005). Motion modeling for on-line locomotion synthesis. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 29–38.

Kwon, W., Kim, H. K., Park, J. K., Roh, C. H., Lee, J., Park, J., kuk Kim, W., and Roh, K. (2007). Biped humanoid robot mahru iii. *7th IEEE-RAS International Conference on Humanoid Robots*, pages 583 – 588.

Laszlo, J. F., van de Panne, M., and Fiume, E. (1996). Limit cycle control and its application to the animation of balancing and walking. *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 155 – 162.

Lau, M. and Kuffner, J. J. (2005). Behavior planning for character animation. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 271 – 280.

Lau, M. and Kuffner, J. J. (2006). Precomputed search trees: Planning for interactive goal-driven animation. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 299 – 308.

Lee, J., Chai, J., Reitsma., P. S. A., Hodgins, J. K., and S.Pollard, N. (2002). Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH)*, 21(3):491 – 500.

Lee, J. and Lee, K. H. (2004). Precomputing avatar behavior from human motion data. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 68(2):79–87.

Lee, Y., Kim, S., and Lee, J. (2010). Data-driven biped control. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 129.

Li, Y., Wang, T., and Shum, H.-Y. (2002). Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics (SIGGRAPH)*, 21(3):465 – 472.

Liu, C. K. (2009). Dexterous manipulation from a grasping pose. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):Article 59.

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

Liu, C. K. and Popović, Z. (2002). Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (SIGGRAPH)*, 21(3):408 – 416.

Liu, L., Yin, K., van de Panne, M., Shao, T., and Xu, W. (2010). Sampling-based contact-rich motion control. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 128.

Lo, W. and Zwicker, M. (2008). Real-time planning for parameterized human motion. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 29–38.

Macchietto, A., Zordan, V., and Shelton, C. R. (2009). Momentum control for balance. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):Article 80.

McCann, J. and Pollard, N. S. (2007). Responsive characters from motion fragments. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):Article 6.

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

Miura, H. and Shimoyama, I. (1984). Dynamic walk of a biped. *International Journal of Robotics Research*, 3(2):60–74.

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

Morimoto, J. and Atkeson, C. G. (2007). Learning biped locomotion: Application of poincare-map-based reinforcement learning. *IEEE Robotics & Automation Magazine*, 14(2):41 – 51.

Morimoto, J., Atkeson, C. G., Endo, G., and Cheng, G. (2007). Improving humanoid locomotive performance with learnt approximated dynamics via guassian processes for regression. In *International Conference on Robotics and Automation (ICRA)*, pages 4234 – 4240.

Morimoto, J., Cheng, G., Atkenson, C., and Zeglin, G. (2004). A simple reinforcement learning algorithm for biped walking. In *International Conference on Robotics and Automation (ICRA)*,

pages 3030 – 3035.

Muico, U., Lee, Y., Popović, J., and Popović, Z. (2009). Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):Article 81.

Mukai, T. and Kuriyama, S. (2005). Geostatistical motion interpolation. *ACM Transactions on Graphics (SIGGRAPH)*, pages 1062–1070.

NASA (2010). Robonaut, `http://robonaut.jsc.nasa.gov/default.asp`.

NaturalMotion (2010). Naturalmotion, `http://www.naturalmotion.com/`.

Ng, A. Y. and Jordan, M. (2000). Pegasus: A policy search method for large mdps and pomdps. In *16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415.

Ng, A. Y. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *17th International Conference on Machine Learning*, pages 663–670.

ODE (2010). Open dynamics engine, `http://www.ode.org/`.

Popović, Z. and Witkin, A. (1999). Physically based motion transformation. *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20.

Pratt, J., Chew, C., Torres, A., Dilworth, P., and Pratt, G. (2001). Virtual model control: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research*, 20(2):129–143.

Pratt, J. and Krupp, B. (2008). Design of a bipedal walking robot. In *Proceedings of the 2008 SPIE*, volume 6962.

Pratt, J. E. and Drakunov, S. V. (2007). Derivation and application of a conserved orbital energy for the inverted pendulum bipedal walking model. In *International Conference on Robotics and Automation (ICRA)*, pages 4653–4660.

Pratt, J. E. and Tedrake, R. (2006). Velocity based stability margins for fast bipedal walking. In *Fast Motions in Biomechanics and Robots*, pages 299–324.

Raibert, M., Blankespoor, K., Nelson, G., Playter, R., and the BigDog Team (2008). Bigdog, the rough-terrain quadruped robot, `http://www.bostondynamics.com/img/BigDog_IFAC_Apr-8-2008.pdf`.

Raibert, M. H. (1986). *Legged robots that balance*. Massachusetts Institute of Technology, Cambridge, MA, USA.

Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, 25(4):349 – 358.

Ramachandran, D. and Amir, E. (2007). Bayesian inverse reinforcement learning. In *20th International Joint Conference on Artificial Intelligence*, pages 2586–2591.

Reitsma, P. S. A. and Pollard, N. S. (2007). Evaluating motion graphs for character animation. *ACM Transactions on Graphics*, 26(4):Article 18.

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

Russell, S. (1998). Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 101–103.

Safonova, A. and Hodgins, J. K. (2007). Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):Article 106.

Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (SIGGRAPH)*, 23(3):514 – 521.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., and Fujimura, K. (2002). The intelligent asimo: system overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2478–2483.

Sharon, D. and van de Panne, M. (2005). Synthesis of controllers for stylized planar bipedal walking. In *International Conference on Robotics and Automation (ICRA)*, pages 2387 – 2392.

Shinjiro, S., Kaufman, A., and Pai, D. K. (2008). Musculotendon simulation for hand animation. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):Article 83.

Shiratori, T., Coley, B., Cham, R., and Hodgins, J. K. (2009). Simulating balance recovery responses to trips based on biomechanical principles. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 37–46.

Sims, K. (1994). Evolving virtual creatures. *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22.

Sok, K. W., Kim, M., and Lee, J. (2007). Simulating biped behaviors from human motion data. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):Article 107.

Sok, K. W., Yamane, K., Lee, J., and Hodgins, J. (2010). Editing dynamic human motions via momentum and force. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Srinivasan, M. and Ruina, A. (2006). Computer optimization of a minimal biped model discovers walking and running. *Nature*, 439(7072):72 – 75.

Sunada, C., Argaez, D., Dubowsky, S., and Mavroidis, C. (1994). A coordinated jacobian transpose control for mobile multi-limbed robotic systems. In *International Conference on Robotics and Automation (ICRA)*, pages 1910–1915.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Takenaka, T., Matsumoto, T., and Yoshiike, T. (2009). Real time motion generation and control for biped robot, first report: Walking gait pattern generation. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*.

Taylor, G. (2008). Compact modeling of high-dimensional time-series. Depth Oral Paper, University of Toronto.

Tedrake, R., Zhang, T., and Seung, H. (2004). Stochastic policy gradient reinforcement learning on a simple 3d biped. In *Proc. IROS*, volume 3, pages 2849 – 2854.

Treuille, A., Lee, Y., and Popovic, Z. (2007). Near-optimal character animation with continuous control. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):Article 7.

Tsai, Y.-Y., Lin, W.-C., Cheng, K. B., Lee, J., and Lee, T.-Y. (2009). Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics*, 99(1):325 – 337.

van de Panne, M. (1997). From footprints to animation. *Computer Graphics Forum*, 16(4):211–223.

van de Panne, M. and Fiume, E. (1993). Sensor-actuator networks. *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 335 – 342.

van de Panne, M., Kim, R., and Flume, E. (1994). Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94*, pages 208–215.

Wampler, K. and Popović, Z. (2009). Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):Article 60.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2005). Gaussian process dynamical models. In *Proc. Neural Information Processing Systems Conf.*, pages 1441–1448.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2009). Optimizing walking controllers. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 28(5):Article 127.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2010). Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 73.

Wei, X. and Chai, J. (2010). Videomocap: Modeling physically realistic human motion from monocular video sequences. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 42.

Witkin, A. and Kass, M. (1988). Spacetime constraints. *SIGGRAPH '88 Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159 – 168.

Wooten, W. (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*.

Wu, C.-C. and Zordan, V. (2010). Goal-directed stepping with momentum control. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Wu, J. and Popovic, Z. (2010). Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 72.

Yamane, K., Kuffner, J. J., and Hodgins, J. K. (2004). Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (SIGGRAPH)*, 23(3):532 – 539.

Yamane, K. and Nakamura, Y. (2003). Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics*, 9:352–360.

Ye, Y. and Liu, C. K. (2008). Animating responsive characters with dynamic constraints in near-unactuated coordinates. *ACM Transactions on Graphics (SIGGAPH Asia)*, 27(5):Article 112.

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

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*, pages 445 – 449.

Yin, K., Coros, S., Beaudoin, P., and van de Panne, M. (2008). Continuation methods for adapting simulated skills. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):Article 81.

Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: simple biped locomotion control. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):Article 105.

Yoshida, E., Belousov, I., Esteves, C., and Laumond, J. (2005). Humanoid motion planning for dynamic tasks. In *International Conference on Humanoid Robots*, pages 1 – 6.

Zeglin, G. and Brown, B. (1998). Control of a bow leg hopping robot. In *International Conference on Robotics and Automation (ICRA)*, pages 793–798.

Zhao, L. and Safonova, A. (2008). Achieving good connectivity in motion graphs. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 127–136.

Zordan, V. B. and Hodgins, J. K. (2002). Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 89–96.

Zordan, V. B., Majkowska, A., Chiu, B., and Fast, M. (2005). Dynamic response for motion capture animation. *ACM Transactions on Graphics (SIGGRAPH)*, 24(3):697 – 701.