Towards Adaptive and Directable Control of Simulated Creatures

by

Yeuhi Abe

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

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2007

Author .....
Department of Electrical Engineering and Computer Science
Febr[illegible], 2007

Certified by .....
Jovan Popović
Associate Professor
or

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Towards Adaptive and Directable Control of Simulated Creatures

by

Yeuhi Abe

Submitted to the Department of Electrical Engineering and Computer Science
on February 2, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Interactive animation is used ubiquitously for entertainment and for the communication of ideas. Active creatures, such as humans, robots, and animals, are often at the heart of such animation and are required to interact in compelling and lifelike ways with their virtual environment. Physical simulation handles such interaction correctly, with a principled approach that adapts easily to different circumstances, changing environments, and unexpected disturbances. However, developing robust control strategies that result in natural motion of active creatures within physical simulation has proved to be a difficult problem. To address this issue, a new and versatile algorithm for the low-level control of animated characters has been developed and tested. It simplifies the process of creating control strategies by automatically accounting for many parameters of the simulation, including the physical properties of the creature and the contact forces between the creature and the virtual environment. This thesis describes two versions of the algorithm (one fast and one feature-rich) and the experiments conducted to evaluate its performance. The results include interactive animations of active creatures manipulating objects and balancing in response to significant disturbances from their virtual environment. The algorithm is shown to be directable, adaptive, and fast and to hold promise for a new generation of interactive simulations that feature lifelike creatures acting with the same fluidity and grace exhibited by natural beings.

Thesis Supervisor: Jovan Popović
Title: Associate Professor

Acknowledgments

I would like to thank my advisor, friends and family. Also special thanks to Eugene Hsu, Daniel Vlasic, and Tom Buehler for motion capture and multimedia assistance and to the following people for their valuable comments and proofreads: Jiawen Chen, Marco da Silva, Frédo Durand, Russ Tedrake, and Robert Wang.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interactive computer animation is a powerful medium for entertainment and for the communication of ideas. It is used ubiquitously in education, scientific visualization, computer games, and for training purposes. Despite this, the primary methods for synthesizing animation content are offline and slow and require a high level of specialized skill that few individuals possess. In the case of massive-scale, interactive environments, the demand for content can greatly exceed the availability. This has resulted in a large body of animation research focused on solving, exactly, this problem: how to automatically and quickly synthesize animation from only compact, high-level descriptions.

Dynamic simulation has provided a partial solution to this problem for animation of passive phenomena such as cloth, fluids, and rigid bodies. The motion of these objects can be synthesized automatically by numerically integrating simple differential equations that govern their state in time. This has enabled animation of increasingly complex environments while simultaneously reducing demands on talented, human animators. It is already utilized broadly in games and training systems, where dynamic, truly interactive objects are rapidly displacing static, precomputed motion.

The simulation of active articulated bodies (AABs) such as humans, robots, and animals, however, has lagged behind, preventing automated animation of creatures

that act in concert with their simulated surroundings. Most existing interactive systems tackle this problem in one of two ways. Either they ignore dynamics altogether and blindly replay recorded trajectories, or they switch to passive dynamics and animate creatures as lifeless rag dolls. Unfortunately many actions a character might perform cannot be handled by either approach. Consider, for example, a human character holding one end of a chain while attempting to counteract forces applied at the other end by steadying its hands. The motion of the character should not ignore the dynamics of the chain nor should the motion of the character by that of a lifeless ragdoll. A promising solution to this problem (the one explored in this thesis) is to integrate AABs into their dynamic surroundings by executing control strategies that accomplish given actions.

Previous work has already demonstrated control strategies for AABs performing many actions, including walking, running, diving and swimming. However, wide-spread adoption of these techniques is hindered by the problem of overspecialization: control parameters are tuned to specific motion trajectories, AAB dimensions, and simulation properties. For example, control parameters tuned to balance an upright creature need not succeed on uneven ground, for a creature in a crouched posture, nor for one holding a heavy object, even though the objective of balancing is conceptually similar in all cases. As a result, these techniques typically only work when interacting with a static environment (e.g., flat ground) and cannot adapt to variations in manipulated objects (e.g., objects of different size or dimension) nor to a dynamic environment (e.g., uneven or moving ground). Directing AABs using such techniques is also quite difficult because the only way to command motion is through tuning control parameters, which, as a method for describing motion, is neither compact, high-level, nor easy to understand.

This thesis begins to address these issues by presenting two different algorithms for the interactive control of AABs: *prioritized control* (Chapter 3) and *multiobjective control* (Chapter 4). Both algorithms share much in common. They both reduce overspecialization in control by decoupling the description of control strategies from

the computation of control parameters required to accomplish them. This is done by (1) automatically adapting to the pose, size and weight of AABs and (2) automatically accounting for the variation in constraints imposed by contact with objects and the environment. Both algorithms strike a compromise between several motion objectives at once. These objectives may include balancing, tracking, and reaching with end-effectors. Ultimately, directing AABs is easier than with previous methods because modular motion objectives can be easily developed, adapted, and composed to create many variations of each control strategy, with little additional effort. For example, a control strategy designed for balancing an adult human character can almost automatically adapt to balancing a character with a child's dimensions.

Prioritized control is a special case of multiobjective control which is faster in some case. On the other hand, multiobjective control is the more sophisticated of the two algorithms and it has two key advantages: (1) it handles unilateral frictional contacts with the environment and (2) it allows for soft trade-offs between conflicting motion objectives that are simultaneously active, rather than enforcing strict priority levels.

In the remainder of this thesis, an overview of relevant background material comes first. This includes a theoretical treatment of AAB dynamics when in contact with the environment, a discussion of prior work on control of AABs, and a comparison with other approaches to interactive character animation. Next, both prioritized control and multiobjective control are described in detail along with the results from experiments performed with both algorithms. For prioritized control, the experiments are focused on the animation of human characters performing object manipulation tasks, such as lifting, catching, and throwing objects of various mass and dimensions. For multiobjective control, the experiments involve dynamic balancing while tracking desired motions despite significant disturbances from the surrounding environment. Lastly, the final chapter summarizes findings and contributions, and suggests possible future directions of exploration.

Chapter 2

Background

2.1 Articulated Body Dynamics

For the purposes of animation, active articulated bodies (AABs) are in constant contact with the surrounding environment. Contact occurs, for example, whenever a humanoid character places a foot on the ground or holds an object in its hands. The motion of an AAB in contact with the environment is significantly more complex than its unencumbered motion in free space. This is due to the presence of reaction forces that push on the body at each contact point. For the common case of sustained contact, however, control can exploit the linear relationship between joint torques, reaction forces, and joint accelerations. This relationship can be computed at interactive rates and used to control AABs. In this section we establish our notation by reviewing contact mechanics and the equations of motion for AABs [6, 47].

2.1.1 Contact Mechanics

Contacts with the environment, as shown in Figure 2-1, restrict the relative velocity of each contact point $\boldsymbol{p}_c^{(i)} \in \mathbb{R}^3$, for $i = 1 \ldots m$. In the case of a non-slipping contact, the relative velocity is zero: $\dot{\boldsymbol{p}}_c^{(i)} = \mathbf{0}$. This condition can also be expressed in terms

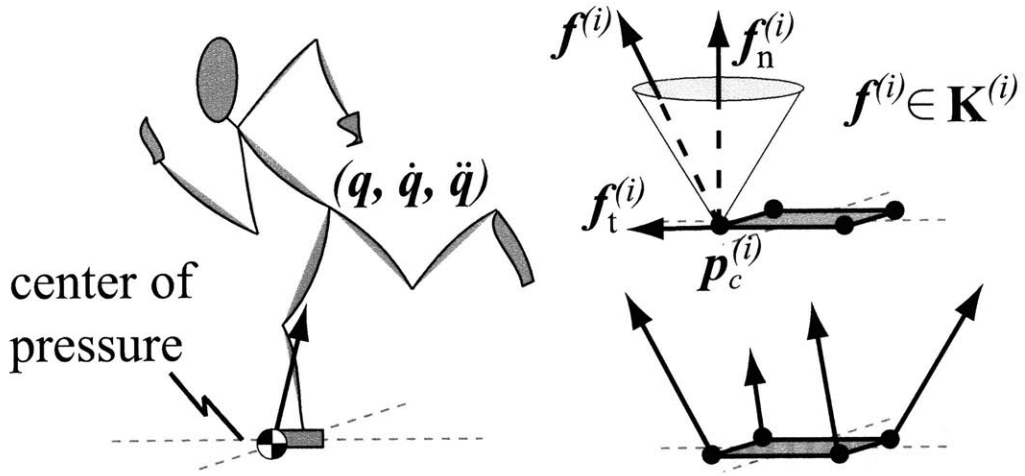

Figure 2-1: Contact dynamics expresses the relationship between the motion $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ of an AAB, its internal torques, and external forces. We model the contact between two surfaces with a set of point contacts $\boldsymbol{p}_{\mathrm{c}}^{(1)} \ldots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

of joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ by using the Jacobian matrix $\mathbf{L}^{(i)} \in \mathbb{R}^{3 \times n}$ to compute the body velocity at the point of contact:

$$\mathbf{L}^{(i)}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{p}}_{\mathrm{c}}^{(i)} = \mathbf{0}. \tag{2.1}$$

A point contact yields a frictional contact force $\boldsymbol{f}^{(i)} \in \mathbb{R}^3$ that prevents geometric overlap by pushing back on the body. Unlike the forces in a joint linkage (bilateral contact), a contact force does not pull the body in case of separation (unilateral contact) implying that its normal component must be positive: $f_{\mathrm{n}}^{(i)} \geq 0$. Coulomb's model of friction limits the tangential component of the contact force: $\|\boldsymbol{f}_{\mathrm{t}}^{(i)}\| \leq \mu f_{\mathrm{n}}^{(i)}$, where $\mu > 0$ is a coefficient of friction at the contact point. We collect these limits into a friction cone $\mathrm{K}^{(i)}$ that restricts the direction and magnitude of the contact force:

$$\boldsymbol{f}^{(i)} \in \mathrm{K}^{(i)} = \left\{ \mathbf{x} \;\middle|\; \|\mathbf{x}_{\mathrm{t}}\| \leq \mu \mathbf{x}_{\mathrm{n}} \right\}. \tag{2.2}$$

By the principle of virtual work, a linear map $\mathbf{L}^{\top} \boldsymbol{f}$ determines the total joint torque by aggregating all contact forces and all Jacobian matrices into one vector $\boldsymbol{f} \in \mathbb{R}^{3m}$

and one matrix $\mathbf{L} \in \mathbb{R}^{3m \times n}$.

2.1.2 Active Articulated Body Dynamics

Conservation of momentum dictates that the total sum of contact forces equals the total change in linear and angular momentum. In other words, in the absence of contact forces, it is impossible for an active body to control the location of its center of mass (COM). An active body propels itself using joint torques $\boldsymbol{u} \in \mathbb{R}^{n-6}$. These torques affect only internal joints $\boldsymbol{q}_1 \in \mathbb{R}^{n-6}$ leaving the global position and orientation of the body $\boldsymbol{q}_2 \in \mathbb{R}^6$ as unactuated degrees of freedom. Using this same separation on the equations of motion produces two sets of equations, with and without actuation:

$$\mathbf{M}_1(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{L}_1^\top(\boldsymbol{q})\,\boldsymbol{f} = \boldsymbol{u} \tag{2.3}$$

$$\mathbf{M}_2(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{L}_2^\top(\boldsymbol{q})\,\boldsymbol{f} = \mathbf{0}. \tag{2.4}$$

The first two terms in both equations combine the inertial and gravitational forces on the body. The two equations summarize the main challenge of active body control with frictional contacts: the dimension of the quantity we need to control $\boldsymbol{q}$ exceeds the dimension of torques $\boldsymbol{u}$ at our disposal. Careful manipulation of contact forces $\boldsymbol{f}$ is the only way to accomplish a specific objective, and yet they are restricted by the friction cone: $\boldsymbol{f} \in \mathrm{K} = \mathrm{K}^{(1)} \times \cdots \times \mathrm{K}^{(m)}$.

2.2 Articulated Body Control

In recent years, AAB control has found its way into commerical animation systems (e.g., *www.naturalmotion.com*) that come packaged with pre-tuned, proprietary control strategies. Many of these systems are probably akin to the earier work of Raibert and Hodgins [38], which relied on spring-damper mechanisms to compute torques for online control. This approach led to some of the most dramatic simulations of active

bodies [19, 52, 8]. However, it required significant tuning of individual rest lengths and spring constants.

Manual tuning can be reduced to some extent with dynamic scaling laws and automated search, but this reaches its limits when adapting to new environments, mass distributions, and other variations [18]. The control algorithms described in this thesis enable modular specification of general control policies for the case of sustained contact. Instead of designing and tuning spring-based dampers for each joint, the specification of control policies is divorced from the computation of required control torques. Hence, the algorithms adjust more easily to new situations and different environments.

Another successful approach to controlling AABs is limit-cycle control. It tracks periodic motions by computing control perturbations needed to return the present motion back to the desired limit cycle (i.e., limit-cycle control strategy or periodic motion data) [29]. Instead of relying on explicit models of contact dynamics, limit-cycle control approximates the Poincaré return map. The advantage of such an approach is that it also incorporates the effects of general frictional contacts (e.g., breaking, slipping, and colliding) into the execution of control policies: a difficult problem that we do not address in this thesis. One major drawback, however, is that it only works for periodic motions such as walking. Almost all the motions we consider in this thesis are non-periodic and, thus, limit-cycle control does not apply to them.

Other tracking alternatives have also been proposed to create dynamically responsive motions from kinematic or preplanned trajectories [56, 55, 57]. These approaches scale spring constants by inertial parameters or feed-forward torques magnitudes to reduce the difficulty of tuning parameters, but none explicitly account for contact dynamics. Some techniques have pursued a hybrid alternative instead, accounting for some dynamic parameters with the goal of generating dynamically feasible *motions* instead of control torques [13, 54, 28]. Theses approaches suffer from the drawback that they do not easily integrate with general purpose simulators of rigid bodies. In contrast, the algorithms we describe easily animate AABs in complex interactions

with many moving objects using any rigid-body simulator.

At the early stages of development, the prioritized control algorithm presented in this thesis was inspired by different prioritized control of articulated bodies developed by Khatib and colleagues [22]. Similar to our approach, the so called "operational space" formulation simplifies control of complex humanoid robots with many degrees of freedom by decoupling the control needed to accomplish a task from the control of task-redundant degrees of freedom. However, the approach taken in this thesis is easier to implement and better at handling closed-loop constraints and frictional contacts, both necessary for animation purposes.

2.3 Character Animation

There are many approaches to animating characters besides dynamic simulation. But regardless of the approach, it must account for both the dynamics and the kinematics of moving characters because static considerations alone will not generate lifelike motion [30]. For example, if dynamic considerations are ignored, lifting heavy objects will look identical to lifting light objects despite the fact that the heavier object should require increased effort and a different motion.

Motion learning approaches resolve this problem with data sets that explore variation in task performance [39, 35, 26]. Although this is effective when tasks can be restricted to small, well-sampled behaviors, more general tasks require solutions to increasingly difficult or ill-posed machine-learning problems. To extend the range of a limited data set, current interactive applications rely on motion-editing tools that approximate dynamics considerations with temporal smoothness objectives [3, 51, 14, 5] This is because dynamically consistent editing tools have not yet been designed for interactive use [37, 32, 45]. Ultimately, temporal smoothness is a poor substitute for the true dynamics exhibited by properly controlled AABs within a physical simulation.

Other approaches execute preplanned motions in simulation using joint-space PD

control, which tracks joint trajectories [56, 55]. Joint-space control has also been successful in animation of lifelike locomotion and other activities [46, 38, 19, 16, 29, 8]. However, joint-space control techniques do not allow for precise control of the motion or forces applied to manipulated objects. Our control algorithms eases the animation of dynamic manipulation by explicitly accounting for object dynamics and supporting intuitive descriptions of motion and force limits directly in the Cartesian space of the objects being manipulated. We call this Cartesian-space control.

Cartesian-space control of end-effectors allows for compact motion description because it commands only the precise details of important points on the body such as feet and hands. In general, compact task descriptions are preferred in both manual [30] and automatic [25] task planning because they suppress irrelevant aspects of task execution. For example, inverse kinematics is often used, to infer full postures from a compact description of the motion of hands, making it easier to reuse performances by different (e.g., shorter or longer-armed) characters [53]. Achieving lifelike postures, however, requires that such algorithms either incorporate recorded motion data or leverage prior results from neurophysiology or other studies of natural motion [25, 40, 15, 53]. The control algorithms described in this thesis address this problem by supporting multiple motion objectives in a strictly prioritized or a weighted fashion. This has been explored before in the kinematic setting [4], however, this thesis is the first to explore its potential for the control of AABs, for animation purposes.

Chapter 3

Prioritized Control

Prioritized control computes the joint torques that cause animated characters to accomplish desired manipulations (Figure 3-1). The algorithm can be used within physical simulation to author new motions or to execute flexible motion control strategies interactively. It is particularly suitable for interactive use because it supports compact task descriptions and the prioritization of conflicting tasks, both of which can simplify the way that motion is commanded. For example, the control algorithm can be used to compactly describe the motion of the character's hands, by specifying a couple goal positions in Cartesian space, while automatically favoring natural postures (infered from recorded motion data) for the rest of the body.

Due to an analytic solution, prioritized control can be executed quite fast, but it suffers from an inability to model unilateral contacts (a flaw which is remedied by multiobjective control in the next chapter). Points of contact with objects and the environment must be fixed in place. For example, hands must firmly grasp manipulated objects and the contact between the feet and the ground must act as if the feet are held to the ground with glue. This can lead to unnatural motion if postures are not monitored to prevent it. Nonetheless, we have found that such bilateral contact can serve as a suitable approximation to unilateral contact in some cases. For example, in the results section, we demonstrate how prioritized control can incorporate

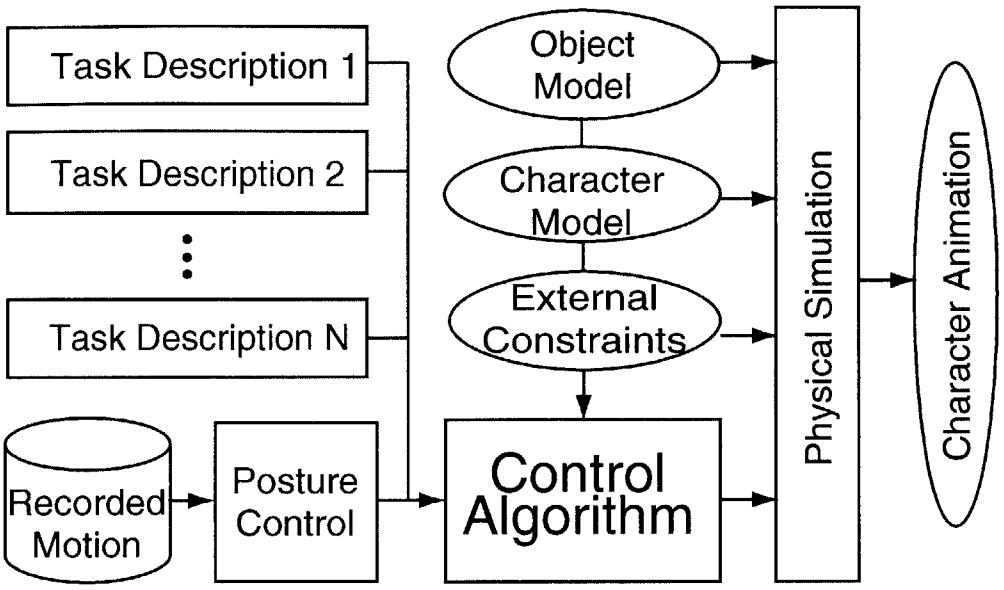

Figure 3-1: Prioritized control algorithm incorporates recorded motion data to accomplish multiple tasks such as lifting, reaching, and throwing within interactive physical simulations.

high-quality motion data to guide complex characters, with many degrees of freedom, through lifelike portrayals of common manipulation tasks. Despite the unrealistic fixed footing, in practice the animations are compelling and lifelike.

3.1 Algorithm

Here we derive the basic control algorithm for unconstrained, open-loop structures before extending it to the most practical case: constrained dynamics with unactuated degrees of freedom. The end result is a procedure that transforms the complex nonlinear dynamics of AABs in contact with the environment into simple second-order linear systems whose intuitive control is explained in Section 3.2.

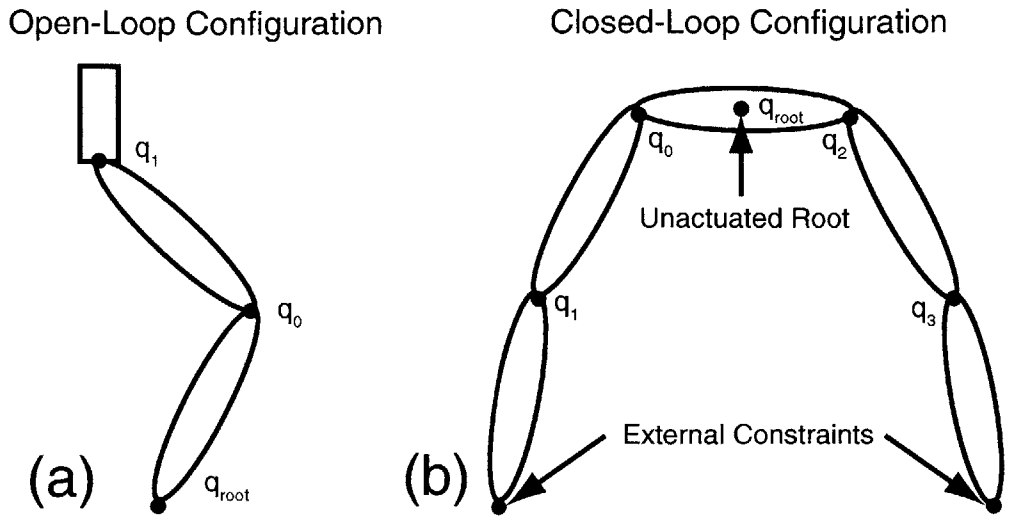

Figure 3-2: In the unconstrained, open-loop configuration (a) the shape is fully described by independent coordinates $\boldsymbol{q}$, whereas in the constrained, closed-loop configuration (b) no set of independent coordinates can describe the shape, so constraints must be handled in the dynamics.

3.1.1 Unconstrained Dynamics

The dynamics of animated characters is modeled as a set of rigid body limbs constrained by a set of joints that link the limbs into a core body structure. When this structure forms a tree graph, also called an open-loop configuration, the pose of the character can be described by a set of independent joint variables (see Figure 3-2). These independent coordinates $\boldsymbol{q}$ allow for the dynamics of the character to be expressed in a standard numerical form:

$$\boldsymbol{u} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{3.1}$$

where $\mathbf{M}$ is the joint-space inertia matrix and $\boldsymbol{n}$ is a nonlinear function of all acceleration-independent terms that computes the gravitational, centrifugal and Coriolis forces [10]. (A derivation of 3.1 can by found in appendix A.) Physical simulations can evaluate and integrate these equations with one of several efficient algorithms, but to animate active characters a control algorithm is still required to supply the joint torques $\boldsymbol{u}$ needed to accomplish desired tasks.

Exact Linearization

Inverse dynamics simplifies design of control algorithms by compensating for complex nonlinear dynamics. The key idea is to transform the nonlinear equations of motion into a linear, second-order system. For example, by choosing joint torques of the form $\boldsymbol{u} = \mathbf{M}\boldsymbol{u}^* + \boldsymbol{n}$, the nonlinear Equation (3.1) is transformed into a set of linear, uncoupled second-order equations, $\ddot{\boldsymbol{q}} = \boldsymbol{u}^*$. This transformation drastically simplifies systematic computation of command torques $\boldsymbol{u}^*$ needed to accomplish joint-space tasks such as tracking procedurally generated trajectories [24] or recorded motion data [55]. Manipulation tasks, however, are not easily described in joint space.

Cartesian coordinates, relative to the needed body part, can be used to intuitively describe manipulation tasks. It is possible to support such descriptions using inverse kinematics, but this approach ignores the dynamics of the task. Instead, our approach applies inverse dynamics in the Cartesian space to directly and intuitively control the task-space dynamics of manipulation tasks. We refer to this as task-space control. Given a differentiable expression $\mathbf{x}_1(\boldsymbol{q})$ for the position (or orientation) of some body part, we can compute its velocity $\dot{\mathbf{x}}_1 = \mathbf{J}_1\dot{\boldsymbol{q}}$ and its acceleration $\ddot{\mathbf{x}}_1 = \mathbf{J}_1\ddot{\boldsymbol{q}} + \dot{\mathbf{J}}_1\dot{\boldsymbol{q}}$ as a function of the Jacobian $\mathbf{J}_1 = \mathrm{D}_{\boldsymbol{q}}\mathbf{x}_1$. Combining the expression for task acceleration with Equation (3.1) allows us to express the dynamics in the Cartesian task space:

$$\boldsymbol{\Omega}_1\boldsymbol{u} = \ddot{\mathbf{x}}_1 + \boldsymbol{\Omega}_1\boldsymbol{n} - \dot{\mathbf{J}}_1\dot{\boldsymbol{q}}, \tag{3.2}$$

where $\boldsymbol{\Omega}_1 = \mathbf{J}_1\mathbf{M}^{-1}$ can be thought of as the pseudoinverse of a task-space inertia matrix.

As before, we compensate for nonlinearities by using inverse dynamics to transform task-space dynamics into a set of linear uncoupled equations. Unlike the joint-space control, however, the systems of equations in task-space control is underdetermined requiring that we choose one of many possible torques. For example, the well known operational space formulation uses the pseudoinverse that minimizes the instantaneous kinetic energy [21]. In contrast, our formulation will compute the complement

joint torque $\bar{\boldsymbol{u}}$ to incorporate motion data into control of dynamic manipulations:

$$\boldsymbol{u} = \boldsymbol{\Omega}_1^+(\mathbf{f}_1^* + \boldsymbol{\Omega}_1\boldsymbol{n} - \dot{\mathbf{J}}_1\dot{\boldsymbol{q}}) + \mathbf{P}_1\bar{\boldsymbol{u}}, \tag{3.3}$$

where $\boldsymbol{\Omega}_1^+$ is any generalized pseudoinverse of $\boldsymbol{\Omega}_1$ and $\mathbf{P}_1 = (1 - \boldsymbol{\Omega}_1^+\boldsymbol{\Omega}_1)$ is the projection matrix onto the null space of $\boldsymbol{\Omega}_1$. Applying this joint torque to Equation (3.2), transforms the nonlinear task dynamics into a simple, second-order linear system, $\ddot{\mathbf{x}} = \mathbf{f}_1^*$, which eases description and control of manipulation tasks. The projection matrix ensures that the complement torque does not interfere with the primary manipulation task. Multi-task control, as described next, directs the remaining degrees of freedom to incorporate other tasks that control the posture of the character, for example.

Multi-Task Control

Multi-task control compensates for the nonlinear dynamics in both high priority and low priority tasks, allowing for precise and intuitive control of manipulations and the style with which they are performed. We again use inverse dynamics to linearize the dynamics of secondary tasks, but we cannot use Equations (3.1–3.3) because secondary tasks are affected by the joint torque $\boldsymbol{u}_1 = \boldsymbol{\Omega}_1^+(\mathbf{f}_1^* + \boldsymbol{\Omega}_1\boldsymbol{n} - \dot{\mathbf{J}}_1\dot{\boldsymbol{q}})$ needed to accomplish the primary manipulation task and, also, by the projection matrix $\mathbf{P}_1$ that prevents secondary-task torque $\bar{\boldsymbol{u}}$ from interfering with the higher priority tasks:

$$\boldsymbol{u}_1 + \mathbf{P}_1\bar{\boldsymbol{u}} = \mathbf{M}\ddot{\boldsymbol{q}} + \boldsymbol{n}. \tag{3.4}$$

Depending on the type of secondary task, we can compensate for nonlinear dynamics by applying inverse dynamics in joint space or in task-space. If the task is to track joint values in the motion data, the joint torques are easiest to compute from command torque $\boldsymbol{u}_2^*$ in joint coordinates:

$$\mathbf{P}_1\bar{\boldsymbol{u}} = \mathbf{M}\boldsymbol{u}_2^* + \boldsymbol{n} - \boldsymbol{u}_1. \tag{3.5}$$

Whereas, if the task is more easily expressed in terms of Cartesian coordinates $\mathbf{x}_2(\boldsymbol{q})$, the joint torques are computed from the Cartesian command vector $\mathbf{f}_2^*$:

$$\boldsymbol{\Omega}_2\mathbf{P}_1\bar{\boldsymbol{u}} = \mathbf{f}_2^* + \boldsymbol{\Omega}_2\boldsymbol{n} - \boldsymbol{\Omega}_2\boldsymbol{u}_1 - \dot{\mathbf{J}}_2\dot{\boldsymbol{q}}, \tag{3.6}$$

where $\mathbf{J}_2 = \mathrm{D}_q\mathbf{x}_2$ and $\boldsymbol{\Omega}_2 = \mathbf{J}_2\mathbf{M}^{-1}$, analogous to expressions in the primary-task control.

The derivation of both equations is analogous to the exact linearization of primary-task dynamics. This clarifies that the joint-space control is a special case of task-space control, as seen by using the identity matrix for the task Jacobian in Equation (3.6). In both formulations, the singular projection matrix restricts the computed torque $\bar{\boldsymbol{u}}$ to the set that does not interfere with the control of the primary task. In our implementation, we compute such torques with the singularity-robust pseudoinverse [36, 34], which inverts the singular value decomposition of $\boldsymbol{\Omega}_1$ (or $\boldsymbol{\Omega}_2\mathbf{P}_1$) after eliminating singular vectors with small singular values (e.g. less than 0.001 threshold in our implementation). This prevents large torques in singular directions that can result in an unstable simulation.

Recursive application of the same idea extends this control algorithm to multiple tasks. For example, additional tasks might limit the range of joint variables [31] or maintain balance [56]. Given a set of Cartesian coordinates $\{\mathbf{x}_1(\boldsymbol{q}), \ldots, \mathbf{x}_n(\boldsymbol{q})\}$ and a set of associated command vectors $\{\mathbf{f}_1^*, \ldots, \mathbf{f}_n^*\}$, the multi-task control computes the joint torque $\boldsymbol{u}_i$ that executes the i-th task at a lower priority than the previous $(i-1)$ tasks:

$$\begin{aligned}
\boldsymbol{u}_i &= \boldsymbol{u}_{i-1} + (\boldsymbol{\Omega}_i\mathbf{P}_{i-1})^+(\mathbf{f}_i^* + \boldsymbol{\Omega}_i\boldsymbol{n} - \boldsymbol{\Omega}_i\boldsymbol{u}_{i-1} - \dot{\mathbf{J}}_i\dot{\boldsymbol{q}}), \\
\boldsymbol{u}_1 &= \boldsymbol{\Omega}_1^+(\mathbf{f}_1^* + \boldsymbol{\Omega}_1\boldsymbol{n} - \dot{\mathbf{J}}_1\dot{\boldsymbol{q}}),
\end{aligned}$$

where $\mathbf{P}_i = (1 - (\boldsymbol{\Omega}_i\mathbf{P}_{i-1})^+(\boldsymbol{\Omega}_i\mathbf{P}_{i-1}))$ and $\mathbf{P}_1 = (1 - \boldsymbol{\Omega}_1^+\boldsymbol{\Omega}_1)$. This iterative algorithm naturally resolves task conflicts by executing lower priority tasks with torques that do not interfere with the higher priority tasks.

Our formulation of multi-task control offers an alternative to the formulation proposed in the robotics literature [22, 42]. The two approaches differ in the formulation of secondary-task dynamics in Eq. (3.6). Unlike the robotics formulation, which requires differentiating the quantity called the task-consistent posture Jacobian $\mathbf{J}_{2|1} = \mathbf{J}_2\mathbf{P}_1$, our approach differentiates only the regular posture Jacobian $\mathbf{J}_2$, as seen in the last term of Eq. (3.6). This difference has a profound impact on the ease of implementation and practical application of multi-task control to animation of dynamic manipulation. Unlike the expression $\dot{\mathbf{J}}_{2|1}\dot{\boldsymbol{q}}$ with the task-consistent posture Jacobian, our expression $\dot{\mathbf{J}}_2\dot{\boldsymbol{q}}$ can be computed simply and efficiently without differentiating the complex projection matrix $\mathbf{P}_1$. Furthermore, it can be shown that both formulations do not interfere with high-priority tasks even as they track secondary tasks as accurately as possible. The difference between the two approaches becomes more pronounced in control of constrained dynamics because the analytic expression for the projection matrix, $\mathbf{P}_1$, becomes more complex, making it harder to compute the time derivative $\dot{\mathbf{J}}_{2|1}$, while our formulation eliminates this step entirely.

3.1.2 Constrained Dynamics

Constrained dynamics emerge whenever a character applies more than one limb to a fixed object in the environment. For example, standing with both feet on the ground establishes contact constraints that relate joint variables of one limb to those of the other. These dependencies make it impossible to describe characters with an independent set of joint variables, as was assumed throughout the previous subsection. Instead, we reformulate our control algorithm to use a set of *dependent* joint variables along with a set of constraint torques $\boldsymbol{u}_c$ that enforce relationships imposed by contact constraints:

$$\boldsymbol{u} + \boldsymbol{u}_c = \mathbf{M}\ddot{\boldsymbol{q}} + \boldsymbol{n}, \tag{3.7}$$

where all expressions retain the meaning from the standard formulation of unconstrained dynamics. The derivation of our control algorithm proceeds by computing

the constraint torques prior to exact linearization of constrained dynamics.

The constraint torques are determined by a set of algebraic equations $\phi(\boldsymbol{q}) = 0$, which may, for example, model non-slipping contact by attaching limbs to objects in the environment. The entire set of constraints determines the structure of the constraint torques by prescribing the valid subspace $\boldsymbol{u}_c = \mathbf{L}^\top \lambda$ as a function of the constraint Jacobian matrix $\mathrm{L} = \mathrm{D}_{\boldsymbol{q}}\phi$. This expression allows for computation of the constraint torques by solving for the coefficients λ in the subspace [10]:

$$\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^\top \lambda = \mathbf{L}\mathbf{M}^{-1}\boldsymbol{n} - \dot{\mathbf{L}}\dot{\boldsymbol{q}} - \mathbf{L}\mathbf{M}^{-1}\boldsymbol{u}. \tag{3.8}$$

Given the expression for constraint torques, the derivation of our control algorithm proceeds as before by applying inverse dynamics to compensate for nonlinear dynamics in joint-space or task-space. For example, the control torques for the primary task $\mathbf{x}_1(\boldsymbol{q})$ are computed from the Cartesian command vector $\mathbf{f}_1^*$ using the following relationship:

$$\boldsymbol{\Omega}_1\boldsymbol{\Phi}\boldsymbol{u} = \mathbf{f}_1^* + \boldsymbol{\Omega}_1\boldsymbol{n} + \boldsymbol{\Omega}_1\boldsymbol{\Gamma}(\dot{\mathbf{L}}\dot{\boldsymbol{q}} - \mathbf{L}\mathbf{M}^{-1}\boldsymbol{n}) - \dot{\mathbf{J}}_1\dot{\boldsymbol{q}} \tag{3.9}$$

where $\boldsymbol{\Gamma} = \mathbf{L}^\top(\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^\top)^{-1}$ and $\boldsymbol{\Phi} = (\mathbf{1} - \boldsymbol{\Gamma}\mathbf{L}\mathbf{M}^{-1})$. This expression highlights the practical benefits of our control formulation (cf. Section 3.1.1). Instead of differentiating the new projection matrix $(1 - (\boldsymbol{\Omega}_1\boldsymbol{\Phi})^+(\boldsymbol{\Omega}_1\boldsymbol{\Phi}))$ as proposed in prior work [22, 42], our multi-task control is just as easily applied to both unconstrained and constrained dynamics.

3.1.3 Unactuated Joints

The joint structure of many animated characters includes passive, unactuated joints. The most common example is the six degree of freedom root joint that determines the global translation and orientation of the character. Unlike an active joint that propels limbs with its torques, the root joint does not apply torques or forces to propel the character directly: instead the global motion arises as a consequence of interaction

with the ground and the environment.

We adjust our control algorithm by defining a selection matrix $\mathbf{S}$ that extracts actuated joints $\boldsymbol{q}_a$ from the full set of joint variables $\boldsymbol{q}_a = \mathbf{S}\boldsymbol{q}$. For example, the $(n-6) \times n$ matrix $S = [\boldsymbol{0} \mid \boldsymbol{1}_{n-6}]$ extracts all but the first six joint variables. Its transpose maps the joint torques into a vector that agrees with the dimension of joint variables, allowing us to rewrite constrained dynamics for characters with unactuated joints:

$$\mathbf{S}^\top \boldsymbol{u} + \boldsymbol{u}_c = \mathbf{M}\boldsymbol{q} + \boldsymbol{n}. \tag{3.10}$$

The remaining steps in the derivation of our control algorithm are analogous to Section 3.1.2.

3.2 Task Description

Compact descriptions, which command only essential details such as hand position or applied force, accelerate animation of manipulation tasks and allow for easy, automated motion specification in interactive applications. Instead of setting and readjusting many keyframes, animators can describe just the required task, adjust a few intuitive parameters, and run a simulation to generate a new motion. Lifelike animations emerge automatically, much like in passive physical simulations, and adapt immediately to changes in the environment (e.g., different object motion or weight) or limitations of the character (e.g., locked joints or muscle strength).

Our control algorithm supports compact task descriptions by decoupling complex non-linear dynamics to allow for simplified motion commands in both joint-space and Cartesian task-space. As in keyframe animation systems, joint-space coordinates ease the description of tasks that require specific joint configurations such as poses from recorded motion data and Cartesian task-space coordinates allow for direct control of body parts needed to manipulate objects. The exact linearization of dynamics explained in the last section transforms the nonlinear problem into a simple second-order

linear system. In this section we rely on this reduction to systematize descriptions of common manipulation tasks.

3.2.1 Manipulation

Our descriptions of manipulation tasks rely on two fundamental control primitives: stabilization, which directs characters towards prescribed values such as desired object locations; and tracking, which follows prescribed trajectories, such as those that describe the desired motion of manipulated objects. Both stabilization and tracking provide a way of choosing the command vector $\mathbf{f}^*$ (c.f. Section 3.1) that will accomplish various manipulation goals. Many other choices of the $\mathbf{f}^*$ are possible, but we have deliberately used simple choices to highlight the functionality of our control formulation, rather than confuse the details with complex motion planning strategies.

Since spatial configurations of manipulated objects are described relative to the global Cartesian coordinate frame, their manipulation is easiest to describe in Cartesian coordinates. We express manipulation tasks in Cartesian (or task-space) coordinates by using forward kinematics to compute the position (or orientation), $\mathbf{x}(\boldsymbol{q})$, of relevant body parts. If a character needs to reach for an object or to carry it to another location, we use stabilization to direct its hands to their desired location $\mathbf{x}_d$. Stabilization creates a motion that progressively eliminates the error between the current and desired configurations, $\mathbf{x}(\boldsymbol{q}) - \mathbf{x}_d$, by utilizing the command vector

$$\mathbf{f}^* = k\big(\mathbf{x}_d - \mathbf{x}(\boldsymbol{q})\big) - 2\sqrt{k}\dot{\mathbf{x}}(\boldsymbol{q}). \tag{3.11}$$

Substituting this command vector into the second-order linear system, described in the last section, reveals a critically damped system whose speed of convergence is controlled by the gain coefficient k. Animators can increase the gain to create stiffer motions that accomplish tasks quickly or decrease it to create more relaxed motions. In our animations, we selected gains manually to showcase relaxed, more reactive animations, but in the future gains could also be set automatically according to

measured human responses.

Tracking is used when more precise execution is required. For example, a character tossing an object must release the object at a prescribed location with a precise velocity. In such a case, we use tracking to direct the character's hands along the trajectory $\mathbf{x}_d(t)$ required to generate the required toss velocity. As in stabilization, tracking eliminates the error between the current and desired trajectories by computing the command force $\mathbf{f}^*$ needed for a critically damped system:

$$\mathbf{f}^* = k\big(\mathbf{x}_d(t) - \mathbf{x}(\boldsymbol{q})\big) + 2\sqrt{k}\big(\dot{\mathbf{x}}_d(t) - \dot{\mathbf{x}}(\boldsymbol{q})\big) + \ddot{\mathbf{x}}_d. \tag{3.12}$$

3.2.2 Force Limits

Force limits restrict the magnitude of applied manipulation forces. This ensures that commands are not accomplished with unrealistic joint torques. For example, a heavy object is lifted slower than a light object because of the limits imposed on the application of the upward force. In nature, force limits are a function of muscle strength, but, in animation, force limits are more intuitively specified in the Cartesian task space. Our control algorithm can be extended to impose such limits by thresholding the task-space forces needed to perform each command.

Given a command vector $\mathbf{f}^*$, we can compute the required task-space force $\mathbf{f}$ using the expression for task-space dynamics in Equation (3.2):

$$\mathbf{f} = (\mathbf{J}\mathbf{M}\mathbf{J}^\top)^{-1}(\mathbf{f}^* + \boldsymbol{\Omega}\boldsymbol{n} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}). \tag{3.13}$$

The task-space force $\mathbf{f}$ should be thought of as the external force that must act, in the absence of internal joint torques, to create the motion commanded by the vector $\mathbf{f}^*$ (Figure 3-3). The task-space force is measured in the usual units of force and its maximum magnitude can be adjusted intuitively to control the strength of manipulations. When the task-space force exceeds a preset value, its thresholded value $\hat{\mathbf{f}}$ can be used in place of the original command vector. If thresholding occurs,

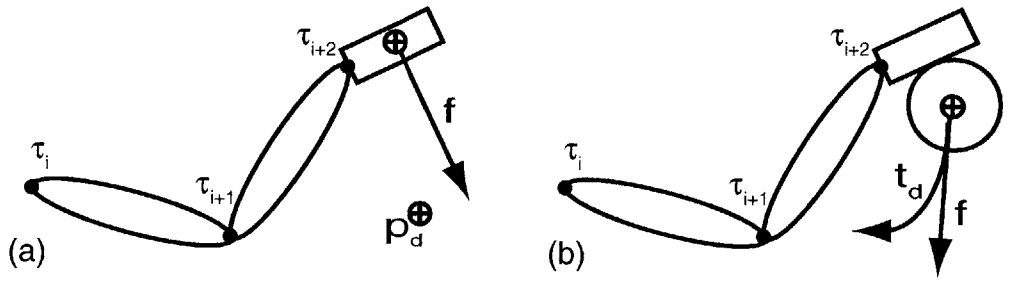

Figure 3-3: Task-space forces guide the hand toward desired position $\mathbf{p}_d$ using stabilization control (a), or move the hand along a specified trajectory $\mathbf{t}_d$, optionally grasping an object (b). For every force $\mathbf{f}$ in task-space, there is an equivalent force $\boldsymbol{u}$ in joint-space that will cause the same motion of the hand and visa-versa.

the Equation (3.13) is inverted to solve for the command vector $\hat{\mathbf{f}}^*$ that corresponds to the thresholded task-space force $\hat{\mathbf{f}}$.

The method we have proposed so far only accounts for force limits in the Cartesian space of the primary task. But in nature force limits are a byproduct of limited muscle strength. Thus, more accurate models should limit forces in the joint-space of characters. Despite this fact, the method we propose has two advantages. First, the animation process is greatly simplified by allowing Cartesian space force limits; It is more intuitive to describe a character's strength by how much the character can lift than by the maximum torque each joint can exert. Second, it is unclear how the motion of the primary task should gracefully degrade when force limits in joint space are reached. Simply clamping the torques will produce unstable motion. Our method always provides modified command vectors that produce manipulation compromise similar to those observed in nature.

3.2.3 Posture

Most manipulation tasks can be accomplished in a number of ways, particularly by complex characters with many degrees of freedom. Although task descriptions command the motion of hands and other body parts, redundancies in body construction allow for variations that are evident in natural motion. The multi-level control for-

mulation allows for systematic description of such variation with posture tasks. As a lower priority task, posture control parameterizes variations without interfering with higher priority manipulation tasks.

Variations depend on many factors including strength, personal preferences, and style. We model these variations by incorporating motion data into a posture task that favors recorded poses. This is implemented as a stabilization task in joint-space, where momentary goal configurations are computed with a nearest-neighbor search through a few seconds of similar motion capture data. The similarity between poses is computed using the horizontal translation- and vertical rotation-invariant distance between synthetic markers affixed to each body part, as first proposed by Kovar and colleagues [27].

Other descriptions of the posture task are also possible. They could be derived from physiological measurements of muscular effort [23, 7] or learned automatically from recorded motion data [15, 35]. Our posture task is a simple variant of the latter choice, aiming to case evaluation of our control technique rather than to improve upon existing posture models.

It should be noted that for realistic motions, posture activity cannot be treated completely independent of the primary task. For example, when lifting a heavy box, a person might choose to do so "with the knees" rather than "with the back" to reduce strain on the muscles. Despite this fact, decoupled motion control has proven a useful abstraction in animation, as demonstrated by the prevalence of inverse kinematic techniques for motion synthesis. As with inverse kinematics, our method depends upon intelligent choices for the posture that compliment the primary task. We leave to future work the development of more sophisticated posture tasks that actively adapt to the goals of the primary task.

3.3 Results

The performance of our control algorithm was evaluated within the Open Dynamics Engine (*www.ode.org*), an open source, high performance library for simulating rigid multibody dynamics. In each experiment, a compact description commands the task for a complex character with 44 degrees of freedom. The control algorithm incorporates postures from supplied motion data to complete the missing details and directs the character in accomplishing each tasks. Collisions and contacts are detected and resolved in the simulation. In particular, grasping and ground contacts are approximated with clamping constraints that affix points on one body to the other. All simulations, including the control computation, run at interactive rates on a 2.8 GHz Pentium 4, with 60 or more updates per second, depending on the task complexity. All animations are available in an accompanying live video.

3.3.1 Chain Interaction.

The chain interaction simulation is a simple demonstration of the immediate benefits gained by incorporating physical effects into animation of manipulation tasks. In this simulation the character attempts to steady its hands while holding onto a serial linkage approximating a chain. Task-space stabilization (c.f. Section 3.2.1) is used to maintain a fixed hand motion as the other end of the chain is tugged and pulled by forces controlled interactively by a mouse-based interface. The secondary posture task keeps the character close to the initial posture. The strength with which the character resists the motion of the chain can be adjusted easily with control of the single gain parameter of the task-space stabilization. Unlike with kinematic techniques, the character reacts to the motion of the chain. In particular, the motion of the legs, while subtle, contributes to a convincing portrayal of this manipulation task.

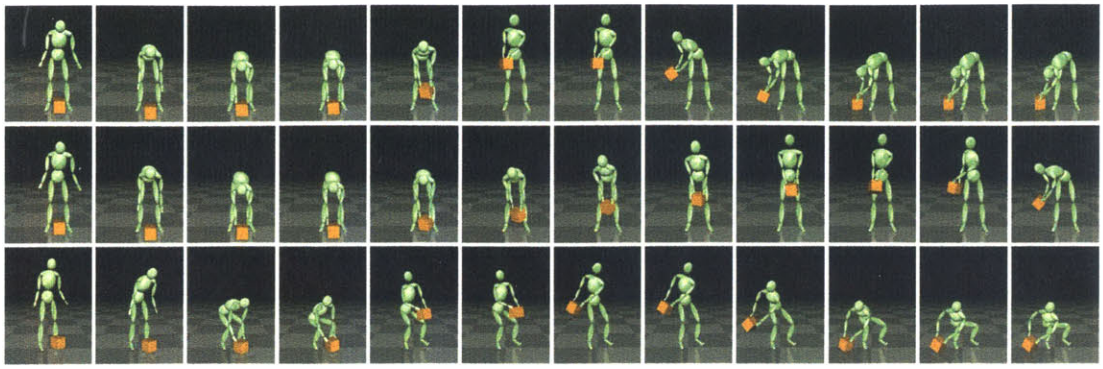

Figure 3-4: Prioritized control directs a real-time simulation of a character to accomplish manipulations, such as displacing a box (top row). Manipulations are compactly described. In the above example, only four Cartesian goal positions are used to describe the motion of the hands and the box. The missing details are filled in with a secondary posture task that incorporates recorded motion postures from a similar performance. The control adapts naturally to changes in the environment. As expected, increasing the weight of the box (second row) produces a slower lift. The performance of the task can also be changed by using a different recorded motion in the posture task (third row).

3.3.2 Lift.

The box lifting simulation demonstrates our algorithm automatically adapting to the weight of objects and incorporating motion data (see Figure 3-4). Stabilization control is used to direct the motion of the hands by specifying keyframes that the hands should pass through. The hands are clamped to the box using simulation constraints between the rigid bodies. Although the control is aware of the box mass (and takes it into account), force limits prevent the character from lifting heavy boxes quickly or even at all. A secondary posture task favors postures from recorded motion data of a similar lifting motion. When we use different recorded data, the performance of the same task description adapts automatically. Instead of lifting "with the back", the character lifts the object "with the knees". This confirms that prioritized control decouples primary and secondary tasks and accomplishes each to the greatest extent possible.

3.3.3 Box Interaction.

The box interaction simulation demonstrates the necessity of dynamic interaction between the character and manipulated objects. The right hand of the character is replaced with a heavy pendulum mass and the desired position of the hand is controlled interactively with a mouse-based interface. The dynamics of the pendulum mass are modeled as that of a body part connected to the arm with an unactuated joint. Stabilization control in task-space is used to bring the arm to the desired position. A secondary posture control references motion capture of a similar motion. This causes the character's posture to vary naturally with the action of the primary control task; the character crouches when the hand is low, stands when the hand is high, and appears balanced even though no explicit balance control is utilized. When the momentum of the pendulum is large, a force limit prevents the character from achieving the desired arm position. However, when the pendulum slows, the force required to achieve the desired position falls below the specified limit and the character can achieve the desired position flawlessly. Note that such precise control is not possible without accounting for the dynamics of the object in the manipulation control. But if, in addition, realistic force limits are not imposed, the character will always achieve the desired hand position perfectly without realistically reacting to the momentum of the pendulum mass. Both force limits and correct dynamics are required to produce believable manipulation.

3.3.4 Catch.

In the ball catching simulation, the character catches balls of different weights, sizes and velocities. Stabilization control is used to position the character's hand approximately where the ball should be caught. When the ball is close to the hand, tracking control is used to match the hand velocity to that of the ball. If contact is detected, the ball is clamped to the hand with a simulation constraint. Finally, stabilization is used to bring the ball back to where the catch was made. The arm configuration

varies naturally with the hand position because the posture task incorporates a short 10-second sequence of arm placement in various catch locations. As the weight of the ball increases, the character reacts naturally. Again, force limits prevent the use of extreme joint torques that might be capable of too quickly stabilizing the position of the hand, regardless of the object weight. Instead, the arm motion slows down the ball before returning to its commanded location.

3.3.5 Catch and Toss.

The catch and toss simulation demonstrate a performance of a more complex manipulation task. The character catches an object before tossing it along the prescribed trajectory. The simulation requires three inputs: the plane in which the character attempts to catch the object, the position and velocity at the point of release, and a motion capture sequence of a similar catch-and-throw motion. The commands in this animation are similar to those in the lifting and catching animations except for the trajectory tracking used to toss the object. The trajectory is a Hermite curve that is fully specified by the initial and final positions and velocities. This parameterization of the curve was chosen for simplicity and looks reasonable for this motion, but it should be noted that the realism of the resulting motion does depend upon the tracking trajectory and, thus, other choice would generate less believable motion. The controller is robust to changes in the velocity and angle of the caught object, the weight, size and shape of the object, and the specified direction and velocity that the object should be thrown. All reasonable settings of these parameters create a plausible motion with different, nonlinear dynamic effects. For instance, if the weight of the object is large, the character will not be able to control the object as accurately, causing collisions between the object and the character, but still tracking the trajectory as closely as possible.

3.4 Discussion

Prioritized control cannot guarantee successful performance of all manipulation tasks. Temporary underactuation (loss of control over some degrees of freedom) will impede manipulation even when it could be accomplished with the remaining degrees of freedom. For example, although a character could jump to reach an object, our control algorithm cannot look ahead to pre-plan the torques needed for such a jump. Although a general solution to underactuated control problems for complex characters is still an open problem, offline optimization has enjoyed some success particularly after simplifying the space of motions [32, 41]. Underactuated control is less critical in authoring applications where animators could be relied upon to provide feasible task descriptions.

The choice of Cartesian-space control eases the description of many manipulation tasks but it also introduces the possibility of artificial algorithmic underactuation. Whenever a jointed structure approaches a singular configuration, the task-space control temporarily loses actuation over some degrees of freedom. This underactuation is artificial because it is strictly a function of the chosen joint-angle parameterization; it never appears in the joint space. In authoring applications, these situations could be avoided with intelligent task descriptions, but a more general solution would impose joint limits in the highest priority task to avoid kinematic singularities [31]. In our work, the posture task serves as a partial substitute to joint limits by keeping the character out of unnatural configurations, but this approach would ultimately fail for extreme postures.

The control algorithm assumes that all contacts are maintained regardless of the applied joint torques. This control strategy is successful for the simulation of some tasks but the control algorithm will need to maintain these contacts explicitly before it can generate animations with realistic locomotion or balance. This is one of the key advantages of the multiobjective control described in the next chapter.

Chapter 4

Multiobjective Control

Like prioritized control, multiobjective control computes the joint torques that cause animated characters to accomplish desired manipulations. However, it has two key advantages over prioritized control. First, it computes a solution to an optimization problem which accommodates unilateral constraints. Such constraints allow for explicit handling of sustained frictional contact with the environment and for limits on the joint torques. Second, the optimization allows for soft trade-offs between simultaneous, conflicting motion objectives, as opposed to the strict priority levels required by prioritized control.

The ability to model unilateral frictional contact is especially important because it occur whenever a creature pushes against the environment and uses the resulting force to control its motion. We refer to this fundamental behavior as standing, noting that it is a precursor to locomotion and other complex behaviors (Figure 4-1). Standing is used more broadly than in its normal connotation. For example, a character performing a handstand or a character bracing itself against a wall with its shoulder is considered to be accomplishing an act of standing.

A key component of multiobjective control is a quadratic program (QP) that maximizes instantaneous performance metrics subject to limits on actuation and contact forces. This theoretical approach is better known in literature on robotic manipu-

lation where frictional contact constrain all interaction between the robot and the object (§2.1). Previous work has applied the QP to motion tracking, but no one has highlighted its promise for control in interactive animation systems or shown how to control active bodies in environments with significant external disturbances.

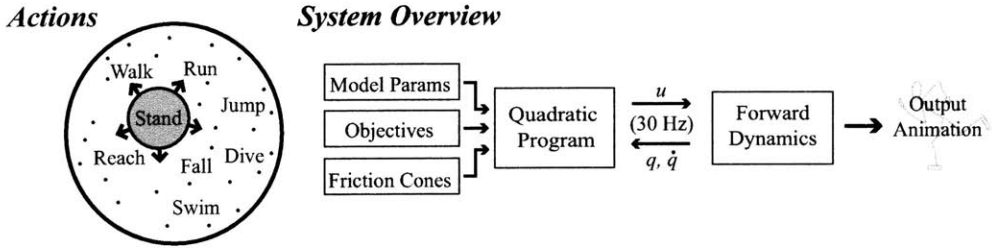

Figure 4-1: Previous control systems demonstrate that many human actions can be simulated. Fundamentally, these actions require careful exploitation of external contact forces during periods of sustained contact. We refer to these periods as "standing". Multiobjective control ensures robust execution of actions while standing. Given a control strategy and physical properties of the body and the environment, our control system uses the current state of the active body $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ to solve a quadratic program that computes the necessary control torques $\boldsymbol{u}$. This allows us to take a fundamental behavior such as standing and expand its range of application to many different scenarios.

Multiobjective control addresses both issues. In the following sections we discuss practical strategies needed to accomplish common control objectives in spite of contact variations caused by significant disturbances (§4.2). In the results section we present lifelike animations of standing characters in challenging physical environments (§4.3). All of which were simulated at interactive rates using a standard rigid-body simulator. These results suggest that multiobjective control may be combined with previously proposed control policies for locomotion and other more complex behaviors and used in the design of a new generation of modular and adaptive control systems (§5).

4.1 Algorithm

The multiobjective control algorithm computes the joint torques that drive the motion of an AAB in simulation. It does so by considering several objectives at once. Each

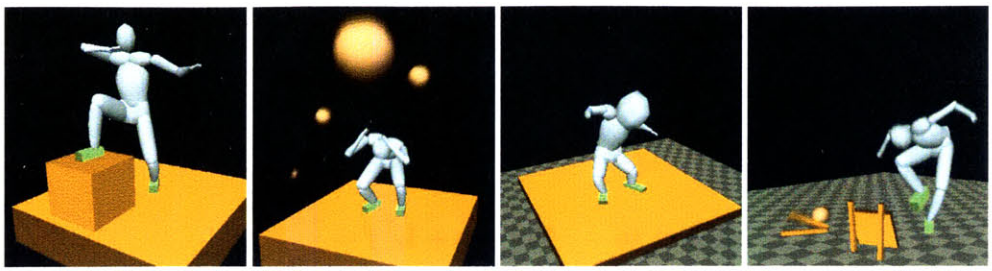

Figure 4-2: Multiobjective control is directable, adaptive, and fast. These images are snapshots taken from interactive simulations driven by our control algorithm. The articulated human body tracks motion capture data and end-effector objectives while maintaining balance. Importantly, our control system automatically adjusts to physical properties of the body, arbitrary frictional contact configurations, and external disturbances.

objective describes a different facet of the desired motion: one objective may insist upon tracking motion data, another may command the location of the center of mass, and yet a third may force the hands to a specific destination. At each instance in time, the conflicts and trade-offs between different objectives are managed by a fast optimization that respects the dynamics of the current contacts and automatically accounts for the physical properties of the AAB. Since speed is a primary requirement of online control, all constraints and objectives are expressed in the form of a quadratic program that can be quickly solved.

In the following subsections, the general form of the optimization problem is described first. Then the details of the QP formulation are discussed, including the exact method for performing control trade-offs between conflicting objectives.

4.1.1 Optimization

Given the current pose $\boldsymbol{q}$ and velocity $\dot{\boldsymbol{q}}$ for the body, the optimization computes joint torques $\boldsymbol{u}$, joint accelerations $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$, and contact forces $\boldsymbol{f}$ that maximize performance

of several objectives $g^{(1)} \ldots g^{(\ell)}$:

$$\begin{aligned}
\min_{\ddot{\boldsymbol{q}},\boldsymbol{f},\boldsymbol{u}} \quad & \{g^{(1)}, \ldots, g^{(\ell)}\} \\
\text{subject to} \quad & \mathbf{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} + \mathbf{L}^{\top}\boldsymbol{f} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \boldsymbol{u} && \text{(4.1a)} \\
& \boldsymbol{f} \in \mathrm{K}, \quad \boldsymbol{u} \in \mathrm{L} && \text{(4.1b)} \\
& \mathbf{L}\ddot{\boldsymbol{q}} + \dot{\mathbf{L}}\dot{\boldsymbol{q}} = \mathbf{0} && \text{(4.1c)}
\end{aligned}$$

In the above, Equation (4.1a) restricts the solution to be consistent with the instantaneous contact dynamics of active articulated bodies. This is a linear constraint on the vector unknowns because the remaining quantities $\mathbf{M}$, $\boldsymbol{n}$, and $\mathbf{L}$ are constant for the current pose and velocity. Equation (4.1b) limits the contact forces and control torques according to current friction cones, K, and constant-bound torque limits, L. Lastly, Equation (4.1c) ensures that accelerations remain compatible with the no-slip contact condition in Equation (2.1).

The last constraint is perhaps the least intuitive of the three. It follows from the linear complementarity condition, which is derived by differentiating the no-slip condition [1]. Its function, however, is best understood by thinking through the contradictory outcome without such a constraint. In that case, joint accelerations are allowed to produce non-zero accelerations at the contact point, which in turn changes the contact forces, even for contact-separating accelerations. We know, however, that contact forces disappear with the separation of contact. Hence, the last constraint ensures that computed torques are consistent with the assumed presence of contact and its contact forces.

4.1.2 Quadratic Program

In practice the implementation approximates the general multiobjective formulation with quadratic programming. This requires choosing quadratic objectives whose

trade-offs are determined by either a strict prioritization or a weighted-sum objective function. Also, nonlinear friction cone constraint are modeled with a conservative polygonal approximation, noting that, if needed, interior point methods could also manage the conical convex constraint in its original form [2].

Quadratic Objectives

In the multiobjective formulation, control strategies are defined by specifying several, possibly conflicting, objectives. For example, different objectives can be used simultaneously to track full body movements and to command positions of hands and feet. The quadratic objectives regulate the values of such kinematic quantities $\boldsymbol{x}(\boldsymbol{q})$ by choosing their accelerations $\ddot{\boldsymbol{x}}(\boldsymbol{q})$ at each moment in time.

The value of each objective $g^{(i)}$ measures the difference between the current $\ddot{\boldsymbol{x}}^{(i)}$ and desired $\boldsymbol{d}^{(i)}$ acceleration:

$$g^{(i)} = \left\| \ddot{\boldsymbol{x}}^{(i)} - \boldsymbol{d}^{(i)} \right\| = \left\| \mathbf{J}^{(i)} \ddot{\boldsymbol{q}} + \dot{\mathbf{J}}^{(i)} \dot{\boldsymbol{q}} - \boldsymbol{d}^{(i)} \right\|, \tag{4.2}$$

where the Jacobian matrix $\mathbf{J}^{(i)}$ describes the linear relationship between joint velocities and velocities of regulated kinematic quantities: $\dot{\boldsymbol{x}}^{(i)} = \mathbf{J}^{(i)} \dot{\boldsymbol{q}}$.

Objectives can be used to reach a desired pose or to track a particular motion trajectory. This feature is particularly useful in animation as it provides a mechanism for incorporating high-quality motion data. If we choose to track motion data $\boldsymbol{m}(t)$, we compute the desired accelerations to encourage a critically damped tracking trajectory:

$$\boldsymbol{d} = k_s\big(\boldsymbol{m}(t) - \boldsymbol{x}\big) + 2\sqrt{k_s}\big(\dot{\boldsymbol{m}}(t) - \dot{\boldsymbol{x}}\big) + \ddot{\boldsymbol{m}}(t), \tag{4.3}$$

where t is the current simulation time and k_s is the tracking stiffness. A high stiffness value produces animations that blindly follow motion data despite external disturbances. We obtain more realistic animations by choosing small stiffness values. Note that such low-stiffness behavior was more difficult to achieve with previous techniques

[56, 55, 54]. The same tracking mechanism can be used to guide the motion of any point on the body, as is needed, to reach for an object and maintain balance.

Control Trade-Offs

Multiobjective control seeks a compromise among various, often conflicting, objectives. One approach to conflict resolution is to identify strict priority levels. A sequence of quadratic programs can then recursively optimize each objective. First, we optimize the most important objective. Next, we constrain its value to the computed optimum and proceed with the optimization of the second most important objective. Strict priority levels ensure that some objectives (e.g., balance) are minimized before others (e.g., reaching). However, we have found that realistic animation requires a more delicate compromise between different objectives.

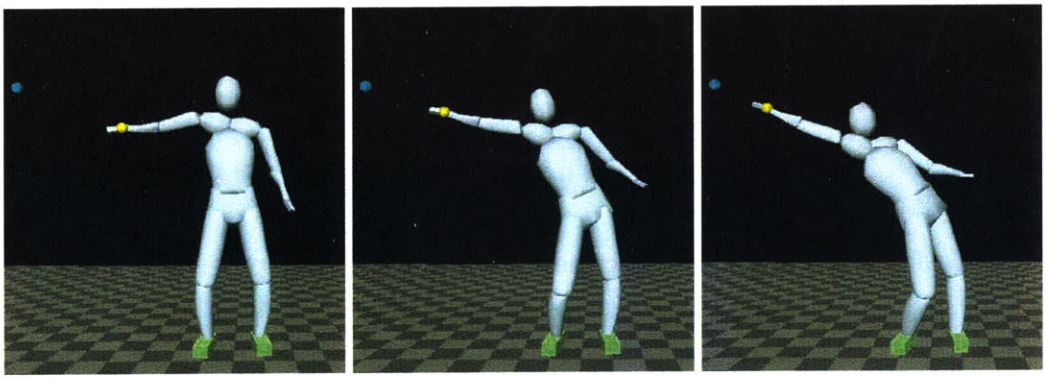

Figure 4-3: The multiobjective formulation allows for explicit control over the trade-offs between different conflicting motion objectives. This figure demonstrates three different trade-offs between the objective of reaching and the objective of remaining upright and balanced. As the weight of the reaching objective is gradually increased, the character assumes a more precarious stance. In the accompanying video, the reaching objective's weight is increased to the point where it outweighs the balance objective, and the character falls over.

We use the weighted-sum objective g to strike a compromise between different control objectives:

$$g = w_1 g^{(1)} + w_2 g^{(2)} + \cdots + w_n g^{(n)} \tag{4.4}$$

We show the effect of different compromises in Figure 4-3. At first, the reaching ob-

jective is given zero weight. Naturally, the arm does not move, but as the importance of the reach increases, the body progressively departs from its balanced stance until the importance of balance is outweighed by the emphasis on reaching and the body falls over. This example clarifies that weighting different objectives is not so much a burden, but an integral and necessary component of a control strategy. Balance, for example, may be a top priority for athletes until they have the opportunity to dive for the ball. Of course, they could also take a step or extend their reach by lifting one leg while balancing on the other. At present, these high-level planning tasks are manually encoded in control objectives, and their weights determine the precise manner in which they are accomplished.

4.2 Practical Control Strategies

In simulation, contact between two objects is neither perfectly detected nor perfectly maintained. Numerical errors due to integration can create variations in detected contact points at almost every time step. External disturbances are even more disruptive. Applying multiobjective control in such an environment requires addressing two major challenges. First, the general theoretical treatment must be complemented with practical strategies that account for frequent contact variation. Second, strategies must be devised that guide the body to positions from which it is capable of accomplishing control objectives such as standing upright to avoid falling.

4.2.1 Stabilizing Contacts

Our theoretical model of contact forces assumes that contacts are maintained. When contacts break, the control must adapt or it will fail. Numerical errors and imprecise object geometry will often create unintended, incidental contact changes. The first step to stabilizing such contacts is to restrict the center of pressure for each contact region to its interior. In our implementation, we restrict the centers of pressure to

scaled versions of the true contact regions (70% of their original size). We also require that contact forces have strictly positive normal components, which is controlled by a weight-dependent threshold. This constraint directs the QP solution to compute torques that push on each contact region and hence discourage incidental changes in contact points, or, in case of small separation, re-establish the contacts in just a few simulation steps.

External disturbances are more disruptive. For larger contact disruptions, we collapse the friction cone $\mathrm{K}^{(i)}$ to disallow tangential contact force and encourage immediate recovery. However, if contact is still not re-established, we remove it from the QP formulation and set its contact force to zero. In that case, we add a new motion objective in a last-ditch attempt to re-establish the contact by guiding the former contact point toward its projection on the external contact surface. In our experiments, these strategies were used to stabilize contacts at each region. More complex behaviors will need to rely on similar strategies to change contact regions intentionally, for example, by taking a step or by reaching for a handle.

4.2.2 Maintaining Controllability

Although humans have an amazing ability to remain standing under many difficult conditions, sometimes we still fall. Equation (2.4) highlights a fundamental physical limitation that makes balancing difficult: the global position and orientation of a body are not directly controlled by joint torques. Humans adapt to this limitation with anticipatory movements just as they brace for the motion of a bus by leaning in the direction of its motion. Our multiobjective control needs a similar mechanism to maintain controllability by guiding the body to configurations from which it is capable of accomplishing several control objectives.

Contact dynamics gives us precise conditions for ensuring controllability (§2.1). The motion of a body $\boldsymbol{q}(t)$ is controllable if and only if there are contact forces $\boldsymbol{f} \in \mathrm{K}$ that

satisfy Equation (2.4):

$$\mathbf{M}_2(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{L}_2^{\top}(\boldsymbol{q})\boldsymbol{f} = \mathbf{0}.$$

This condition is a generalization of two often used alternatives for planar contacts: (1) the center of mass (COM) should project inside the support polygon and (2) the zero-moment point should remain within the support polygon [47]. The condition also incorporates friction and applies to any three-dimensional contact configuration. Unfortunately, direct application of this condition to maximize controllability for a given disturbance is beyond the computational budget of online control systems. The equations are no longer linear because the state of a body changes over time. The most efficient implementation developed by my colleagues (and based on the optimal control framework [50]) required several seconds of computation time, at least an order of magnitude too slow for online control.

A heuristic solution to this problem is to incorporate an objective that guides the COM toward more controllable configurations for most disturbances. The COM for a human standing on flat ground, for example, is usually above the mid-point between the two footprints. In general, this is a controllable configuration for many disturbances because the COM is far from the edges of the contact support. We refer to this objective as the controllability objective because maintaining controllability is its purpose. An even safer strategy might lower the COM (as is the goal of many sumo wrestlers), but this becomes more a cognitive choice than a reflexive maneuver, so we choose to leave this aspect free to be controlled by other objectives.

The key to understanding controllability is to observe that this objective does not prevent falling on its own: the COM can fall to the ground and still be above the mid-point. Instead, falls are prevented with a combination of this and others objectives that prescribe *standing* motions or postures. When conditions of controllability are violated (typically when the COM wanders significantly outside the support polygon) the body falls because it can no longer accomplish the objective of standing. However, we found that our simple strategies worked well even for many significant

disturbances. If need be, more complex strategies for controlling the COM can easily be incorporated using multiobjective control.

4.3 Results

Games and training simulations pose a difficult set of challenges for any animation system: animation must be fast, applicable in many conditions, responsive to disturbances, and easy to direct. Multiobjective control addresses these challenges with a general purpose control system for bodies in sustained frictional contact with their environment. Our experiments show that it meets the demands of interactive systems for the fundamental behavior of standing and suggest that it may provide a strong foundation for the design of even more complex behaviors.

In our testing, we explored a range of different interactive simulations driven by our control system. The supplemental video includes a few typical runs from these experiments:

Sobriety. A human-like character accomplishes a standing upright posture while in uneven contact with a moving platform. It also reaches for its nose as commanded by intuitive objectives describing the desired position of its hands. Both standing and reaching are accomplished despite the significant motion of the platform.

Pelted. The same character can also track motion data. Collisions with other simulated objects generate life-like responses while the motion trajectory remains similar to the data.

Alien. A shorter character accomplishes a standing upright posture on a moving platform, as well as, a new adaptation of the "Pelted" simulation. Although its geometry, weight, and proportions differ from those of the human character, only minor modifications of weight-dependent thresholds are needed to modify the control strategy.

Wall. As directed, our human character places its hand on a nearby wall for additional support on the moving platform. The control system adapts to the non-planar contact configuration and uses its additional leverage to maintain balance despite severe tipping of the platform.

Mishap. The character stands with one leg perched on a flimsy table. When the table suddenly collapses, the character regains its balance in a controlled manner, which we intuitively direct by guiding its foot to a desired location.

We manually modeled the geometry of both characters in our simulations. Their inertial properties were computed automatically using the volume of each limb and standard mass distributions [49]. The motions tracked by our control system were recorded with an optical motion capture system. Forward dynamics with frictional contacts were computed with the Open Dynamics Engine (*www.ode.org*), a general purpose rigid body simulator. The QP problems were solved by the MOSEK software system (*www.mosek.com*), which employs the interior point method to solve convex optimization problems [2].

4.3.1 Direction

Our experiments demonstrate that multiobjective control enables artistic control of active bodies with two familiar animation mechanisms: direct control of poses and end effector positions.

A posture tracking objective allows the user to direct motions via recorded motion-capture sequences. In most of our experiments, we tracked a single recorded posture, but tracking motions is just as easy. Importantly, tracking fast motions, such as dodging incoming objects, is accomplished accurately, but is still "loose" enough to respond interestingly to collisions with other objects (Pelted). Control need not insist on the perfect match between the body and recorded postures. A shorter character, for example, can easily track the recorded trajectory of a full-size human (Alien).

Tracking objectives can also control individual limbs: arms, hands, feet, and so on.

Our experiments include two simple examples. One directs hands to touch the nose (Sobriety) and the other controls the swing leg to direct the look of a balancing maneuver (Mishap). In addition, our control system always relies on direct control of the horizontal location of the center of mass to maintain controllability (§4.2.2).

4.3.2 Adaptation

Multiobjective control also provides a general formulation for mixing several control strategies while automatically adapting to general contact configurations, external disturbances, and physical properties of the character and the environment.

Many control systems assume planar contact with flat ground, which limits possible applications. Multiobjective control manages this special case (Pelted, Alien) but it also handles more general contact configurations such as one foot resting on an object (Sobriety, Mishap) or a hand contact with the wall (Wall).

Many of our experiments feature a character on a moving platform. Under such conditions, our control system maintains controllability by coaxing the center of mass back toward a conservatively chosen position. The corrective motions required to accomplish this, weighted against other active objectives, contribute to the life-like quality of our animations. Although our simple strategies can be improved with further work, our general control formulation can accommodate new strategies once they are available.

A moving platform is only one example of many possible disturbances. The allure of physically based animation is clearly demonstrated by a rich diversity of interactions characters can have with their environment. However, this is only possible if characters can react naturally to arbitrary disturbances. We present a couple of examples (Pelted, Mishap) that, even if not as natural looking as recorded motions, suggest definite progress in this direction. Complex motions, including natural but counter-intuitive balance recoveries, such as lunging in the direction of the fall (Mishap), emerge without explicit modeling.

Simulation	Vars.	Opt. Time (Avg.)	Iterations (Avg.)
Platform	140	13ms	14.5
Pelted	140	13ms	14.5
Sobriety	146	16ms	13.8
Mishap	143	14ms	15.8
Wall	172	29ms	17.7

Table 4.1: The number of variables, average optimization time, and average number of iterations for the multiobjective QP per simulation.

Multiobjective control can even accomplish strategies on bodies with different inertial parameters (Alien). Our shorter character is capable of withstanding significant disturbances by accomplishing general control strategies initially tuned for a taller and heavier character. We only changed the internal weight-dependent threshold for the normal component of contact forces. This highlights a key advantage of our control system: it decouples the description of control strategies from the computation of required torques. Hence, the objectives are independent of mass distribution, model geometry, and contact dynamics.

4.3.3 Speed

The QP control problem is solved 30 times per second of simulation, while we use many more simulation steps in the same interval, between 1000 and 5000. Each solution required around 15 iterations to converge for an average running time of 17 milliseconds. The "Wall" simulation took slightly longer than the others (see Table 4.1) because of the additional hand contact. All simulations were fast enough to allow the entire system (simulation and control) to run at 30 frames per second, or better, on a 2.8 GHz Intel Pentium 4.

4.4 Discussion

The multiobjective approach was inspired by prioritized control of articulated bodies [22]. The principal advantage of such an approach is that it automatically coordinates

multiple objectives, which makes it easy to combine compact task descriptions with the less specific postural objectives gleaned from motion data, as demonstrated by the results. However, prioritized control without unilateral contact constraints assume the existence of contact forces that maintain contact in spite of external motions, as if the bodies were pinned at the contact points. As illustrated in Figure 4-4, pinned contacts produce unrealistic control strategies.

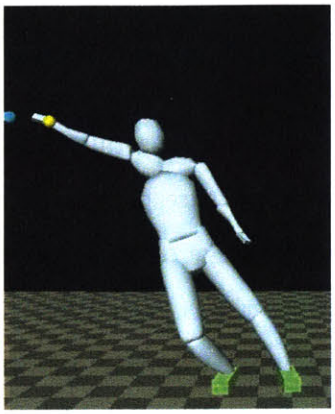

Figure 4-4: This illustration underscores the importance of incorporating ground contact constraints into any control formulation. Ignoring contact dynamics, a character can reach for the object as if his feet were pinned to the ground. With proper contact dynamics and multiobjective control, the character strikes a compromise between reaching and not falling as seen in Figure 4-3.

Ground reference points such as the ZMP provide an alternative to pinning the contact points. The ZMP is a criterion of physical feasibility for bodies in contact with the ground plane [43]. For example, its position outside the contact polygon indicates a physically infeasible motion. The ZMP criterion is sometimes incorrectly defined as a measure of dynamic stability in both graphics and robotics literature. Instead, the ZMP criterion enables successful tracking of *controllable* trajectories by ensuring physically realizable control policies [17]. Hofmann and colleagues, for example, use quadratic programming to restrict the ZMP to remain within the contact polygon [20]. This approach works well for planar contact configurations with infinite friction but not for general three-dimensional contacts with friction [47]. In contrast, our formulation handles arbitrary, non-planar contact configurations with friction.

The theoretical treatment of contacts used by the multiobjective formulation is conceptually similar to the explicit model of contact dynamics used in simulation of rigid bodies [33, 1]. Instead of solving for contact forces that prevent geometric overlap, control torques that are consistent with such forces are computed. According to a

survey by Srinivasa [44], the first control system with an explicit model of contact dynamics appeared in the robotics literature as a solution to multi-fingered manipulation of two-dimensional objects [6]. The control systems proposed in graphics literature, however, did not employ explicit formulations of contact dynamics until Fang and Pollard [9] demonstrated their value to *offline* optimal control. Multiobjective control demonstrates the feasibility and importance of this model for *online* control in interactive animations of active bodies.

Other methods in robotics literature have relied on similar QP formulations for control of walking bipeds [12, 48] without addressing contact variations and significant disturbances. Our work examines its role in the animation of standing AABs. But we specifically emphasize the resilient treatment of disturbances, reasoning that locomotion and more complex behaviors can be robust only after standing is more robust.

Chapter 5

Conclusion

This thesis introduces two control algorithms, prioritized control and multiobjective control, that facilitate the control of complex characters performing lifelike motions within a physical simulation. They support intuitive motion direction through either control of joint angles or end-effector positions and through the ability to execute multiple motion objectives simultaneously. They automatically adjusts to parameters of the character model and of the simulation making it easier to create reusable control strategies that are not overspecialized. Importantly, they are robust to dynamic disturbances in the environment that require significant deviation from specified motion trajectories.

Whereas prioritized control is the faster of the two algorithms, multiobjective control has all the same functionality and two additional advantages. First, it handles unilateral frictional contacts with the environment. This is critical, for example, for realistic motion of characters standing while balancing. Second, it allows for soft trade-offs between conflicting motion objectives that are simultaneously active, rather than enforcing strict priority levels. Although strict priority levels work well for filling in posture details as a secondary objective, my colleagues and I find that trade-offs between objectives such as balancing and reaching are often soft and, thus, should depend upon the intent of the animator.

There are a couple weaknesses to the approach that should be addressed in the future. One weakness is that multiobjective control only models unilateral frictional contacts when they are sustained. The more general case of slipping or breaking frictional contacts also occur in lifelike motion. For example, whenever a baseball player slides into first bases the feet are in slipping frictional contact with the ground. Another weakness is that multiobjective control cannot account for the presence stiff passive elements in a AAB. These include stiffly held joints (e.g., a stiffly held wrist when making a fist) and soft joint limits, as commonly occur in nature. If a character's motion is severely limited such joint limits, the approach in this thesis fails. In many of the simulations from the results section, we automatically detect such situations and revert to passive ragdoll dynamics, but, ultimately, this is a poor substitute for an active control strategy. A third weakness is that multiobjective control is not restricted to smooth joint torques. In some cases, joint torques change rapidly in a manner that could not occur in nature due to the necessary recovery time of muscles. In practice, this can result in strange behaviors, such as non-smooth or jumpy motion, if not carefully avoided.

Although our approach provides a low-level control framework for physically based animation, it does not create motion trajectories from scratch. Rather, it incorporates lifelike trajectories from recorded motion data. As such, it complements kinematic methods that ignore physics but learn from data and other studies of natural motion [25, 40, 15, 53]. By combining the two we can begin to create animation tools that create lifelike motions even in arbitrary, dynamically interactive environments.

In the future, we would like to use multiobjective control to create libraries of modular and reusable control strategies. For example, the simple controllability objective that is employed in the this thesis will not work in all situations. But multiobjective control allows for different objectives to be combined with ease, so as soon as new solutions are devised, they can be easily adapted to different characters and incorporated into existing simulations. Multiobjective control shows great promise for enabling such a modular and adaptive design.

Multiobjective control has potential to enable many applications. In conjunction with high-level action planning and fast rendering techniques, it will eventually allow characters in interactive video games to act in concert with a constantly evolving and dynamic virtual world. In the movie industry, it could potentially allow for automatic animation of massive crowd scene that appear physically realistic even when examined at the level of individual character interactions. It may also serve as an integral part of an animation authoring tool for laymen, since it allows for automatic synthesis of lifelike motion without much manual input. Any or all of these applications are likely candidates for future work.

Appendix A

Equations of Motion

The equations of motion for constrained rigid bodies in an open-loop configuration are:

$$\boldsymbol{u} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{A.1}$$

These equations express the differential relationship between the generalized coordinates, $\boldsymbol{q}$, and joint torques, $\boldsymbol{u}$. This appendix explains the structure of the articulated character model used in our implementation and our approach to the derivation of the equations of motion through applying the principle of virtual work.

A.1 Character Model

The character model is composed of rigid mass segments constrained by joints. The kinematic frame for each rigid segment is described by the composition of homogeneous transformations in a tree-like, hierarchal structure. The structure is rooted at the characters hips. The variable transformations, which allow for the motion of the character, are all rotational transformations (i.e., joints), except for one variable translational transformation describing the global position of the character model. Rotational transformations can have from one to three degrees of freedom (DOFs) depending upon the type of joint they model. 1-DOF rotations are represented in-

ternally by a scalar value representing a rotation about a fixed axis, 2-DOF rotations are represented by the concatenation of 1-DOF transformations, and 3-DOF rotations are represented by normalized quaternions. The root transformation of the hierarchy has 6 spatial degrees of freedom represented by the concatenation of a translation, $T_0(\boldsymbol{q}_{t0})$, and a 3-DOF rotation, $R_0(\boldsymbol{q}_{r0})$, (7 variables total, 3 translation + 4 quaternion). The transformation of child frame r_i w.r.t. its parent frame r_{i-1}, is given by a static translational transformation, T_i, followed by variable joint rotational transformation, $R_i(\boldsymbol{q}_i)$. In the above, $\boldsymbol{q} = [\boldsymbol{q}_{t0}^T \ \boldsymbol{q}_{r0}^T \ \ldots \ \boldsymbol{q}_i^T \ \ldots \ ]^T$.

A.2 Spatial Quantities

Associated with each variable transformation is a subspace, S_i, of $\mathbb{R}^6$ describing the mapping betweem joint velocities and spatial velocity. (See Featherstone [11] for more details.) S_i can be interpreted as a mapping from generalized forces to spatial forces. In other words, $S_i\dot{\mathbf{q}}_\mathbf{i} = \hat{\mathbf{v}}$, where $\dot{\mathbf{q}}_\mathbf{i}$ is the joint velocity and $\hat{\mathbf{v}}$ is the spatial velocity of frame r_i w.r.t. its parent frame. Alternatively, S_i can be viewed as the subspace in which the joints are unconstrainted and actuated. In other words, $S_i\gamma_\mathbf{i} = \hat{\mathbf{f}}_\mathbf{i}$, where $\gamma_\mathbf{i}$ is the generalized joint force and $\hat{\mathbf{f}}_\mathbf{i}$ is an equivalent spatial force applied at the origin of the frame r_i.

Examples of S_i are:

y-axis rotation:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

SO(3) rotation:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3 DOF translation :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

In each frame of the hierarchy, we specify the orientation of rigid body b_i with homogeneous transformation matrix, M_i. We can express the orientation of b_i in any frame f_j $(j \leq i)$ as $T_j^i = T_j R_j T_{j+1} R_{j+1} \ ... \ T_i R_i M_i$. We can also kinematically relate the spatial velocity, $\hat{\mathbf{v}}_{b_i}$, of b_i in its local frame to the joint velocity, $\dot{\mathbf{q}}_\mathbf{j}$, by

$$\hat{\mathbf{v}}_{b_i} = \hat{X}_j^i S_j \dot{\mathbf{q}}_\mathbf{j},$$

where

for $j \leq i$:

$$\hat{X}_j^i = \begin{bmatrix} (R_j^i)^T & -(R_j^i)^T \mathbf{d}_j^i \times \\ 0 & (R_j^i)^T \end{bmatrix}$$

for $j > i$: $\hat{X}_j^i = [0]$

$$T_j^i = \begin{bmatrix} R_j^i & \mathbf{d}_j^i \\ 0^T & 1 \end{bmatrix}$$

$$\mathbf{d}_j^i \times = \begin{bmatrix} 0 & d_z & -d_y \\ -d_z & 0 & d_x \\ d_y & -d_x & 0 \end{bmatrix}$$

$\hat{X}_j^i$ is the adjoint matrix which transforms spatial quantities (i.e. velocity) from frame j to i.

Using the defined quantities above, the aggregate Jacobian of the articulate bodies, $\{b_i | 0 \leq i \geq n\}$, is given by:

$$J_b = \begin{bmatrix} \hat{X}_0^0 S_{t0} & \hat{X}_0^0 S_{r0} & [0] S_1 & \dots & [0] S_n \\ \hat{X}_0^1 S_{t0} & \hat{X}_0^1 S_{r0} & \hat{X}_1^1 S_1 & \dots & [0] S_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{X}_0^n S_{t0} & \hat{X}_0^n S_{r0} & \hat{X}_1^n S_1 & \dots & \hat{X}_n^n S_n \end{bmatrix}$$

The aggregate Jacobian relates the instantanious velocity of the generalized coordinate, $\dot{q}$, to the instantanious motion of the rigid bodies. Since each S_i will have between 1 and 3 dimensions depending upon the joint types, J_b, has dimension $6n \times |\dot{\mathbf{q}}|$.

For each rigid body b_i there is an associated 6×6 spatial inertial tensor

$$\hat{I_{b_i}} = \begin{bmatrix} diag(m_{b_i}) & [0] \\ [0] & I_{b_i} \end{bmatrix},$$

where $diag(m_i)$ is a 3×3 diagonal matrix with the mass of the body on the diagonal and I_{b_i} is 3×3 rotational inertial tensor, in the frame of the body. The $6n \times 6n$ aggregate inertial tensor of the articulated body is,

$$\hat{I_b} = \begin{bmatrix} \hat{I_{b_0}} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \hat{I_{b_n}} \end{bmatrix}$$

.

We also define the spatial transformation matrix R_b that transforms the spatial velocities of the bodies from their local orientation to the orientation of the inertial frame:

$$R_b = \begin{bmatrix} \hat{R}_0^0 & [0] & \dots & [0] \\ [0] & \hat{R}_0^1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & [0] \\ [0] & \dots & [0] & \hat{R}_0^n \end{bmatrix},$$

where

$$\hat{R}_j^i = \begin{bmatrix} R_j^i & [0] \\ [0] & R_j^i \end{bmatrix}.$$

A.3 Derivation from Virtual Work

Using the above, the kinetic energy, e_k, of the system can be written,

$$e_k = (1/2)\hat{\mathbf{v}}_\mathbf{0}^T \hat{I_0} \hat{\mathbf{v}}_\mathbf{0}$$

where $\hat{\mathbf{v}}_\mathbf{0}$ is the vector of spatial velocities in the inertial frame.

Observing that e_k is a positive definite function of $\hat{\mathbf{v}}_\mathbf{0}$ and differentiating w.r.t. $\hat{\mathbf{v}}_\mathbf{0}$, we obtain the spatial momentum of the bodies,

$$\begin{aligned}
\mathbf{l}_0 &= \hat{\mathbf{v}}_0^T \hat{I_0} \\
&= (\hat{\mathbf{v}}_\mathbf{b}^T R_b^T)(R_b \hat{I_b} R_b^T) \\
&= \hat{\mathbf{v}}_\mathbf{b}^T \hat{I_b} R_b^T,
\end{aligned}$$

where $\hat{\mathbf{v}}_\mathbf{b} = [\hat{\mathbf{v}}_{\mathbf{b_0}}^T \ldots \hat{\mathbf{v}}_{\mathbf{b_i}}^T \ldots \hat{\mathbf{v}}_{\mathbf{b_n}}^T]^T$ is the vector of spatial velocities in the reference frames of the bodies.

Using $\boldsymbol{u}$ to denoting the vector of joint torques we observe from the principle of virtual work that

$$\begin{aligned}
\boldsymbol{u}^T \cdot \dot{\mathbf{q}} \delta t &= \frac{d\mathbf{l}_0}{dt} \cdot \hat{\mathbf{v}}_0 \delta t \\
&= \frac{d}{dt}(\hat{\mathbf{v}}_\mathbf{b}^T \hat{I_b} R_b^T) \cdot \hat{\mathbf{v}}_0 \delta t \\
&= \frac{d}{dt}(\dot{\mathbf{q}}^T {J_b}^T \hat{I_b} R_b^T) \cdot \hat{\mathbf{v}}_0 \delta t \\
&= [\ddot{\mathbf{q}}^T {J_b}^T \hat{I_b} R_b^T + \dot{\mathbf{q}}^T \dot{J_b}^T \hat{I_b} R_b^T + \dot{\mathbf{q}}^T {J_b}^T \hat{I_b} \dot{R_b^T}] \cdot \hat{\mathbf{v}}_0 \delta t \\
&= [\ddot{\mathbf{q}}^T {J_b}^T \hat{I_b} R_b^T + \dot{\mathbf{q}}^T \dot{J_b}^T \hat{I_b} R_b^T + \dot{\mathbf{q}}^T {J_b}^T \hat{I_b} \dot{R_b^T}] \cdot (R_b J_b \dot{\mathbf{q}}) \delta t,
\end{aligned}$$

thus,

$$\boldsymbol{u} = {J_b}^T \hat{I_b} J_b \ddot{\mathbf{q}} + [{J_b}^T R_b^T \dot{R}_b \hat{I_b} J_b + {J_b}^T \hat{I_b} \dot{J_b}] \dot{\mathbf{q}}. \tag{A.2}$$

This set of linear equations is in the desired form of A.1, which is used throughout this work. More precisely, we have show that $\mathbf{M} = {J_b}^T \hat{I_b} J_b$ and $\boldsymbol{n} = {J_b}^T R_b^T \dot{R}_b \hat{I_b} J_b + {J_b}^T \hat{I_b} \dot{J_b}$.

We can also add an extra term, $\boldsymbol{u}_{ext}$, to accounts for external forces that depend only upon the configuration of the articulated body (e.g., gravity). In general, $\boldsymbol{u}_{ext}$ can be computed in term of the potential energy, k_p, of the system as,

$$\boldsymbol{u}_{ext}(\mathbf{q}) = \frac{dk_p}{d\mathbf{q}} = gravity * \hat{I_b} * J_b * [0,1,0,0,0,0]^T.$$

The resulting equations of motion are then modified to be:

$$\boldsymbol{u} = {J_b}^T \hat{I_b} J_b \ddot{\mathbf{q}} + [{J_b}^T R_b^T \dot{R}_b \hat{I_b} J_b + {J_b}^T \hat{I_b} \dot{J_b}]\dot{\mathbf{q}} + \boldsymbol{u}_{ext}(\mathbf{q}). \tag{A.3}$$

Bibliography

[1] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, Annual Conference Series, pages 223–232. ACM SIGGRAPH, July 1989.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] A. Bruderlin and L. Williams. Motion signal processing. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, Annual Conference Series, pages 97–104. ACM SIGGRAPH, August 1995.

[4] Benoît Le Callennec and Ronan Boulic. Interactive motion deformation with prioritized constraints. In *Symposium on Computer Animation (SCA)*, pages 163–171, July 2004.

[5] Kwang-Jin Choi and Hyeong-Seok Ko. Online motion retargetting. *Journal of Visualization and Computer Animation*, 11(5):223–235, December 2000.

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

[7] Vincent De Sapio, James Warren, Oussama Khatib, and Scott Delp. Simulating the task-level control of human motion: a methodology and framework for implementation. *The Visual Computer*, 21(5):289–302, 2005.

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

[9] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics*, 22(3):417–426, July 2003.

[10] R. Featherstone and D. E. Orin. Robot dynamics: Equations and algorithms. In *International Conference on Robotics and Automation (ICRA)*, pages 826–834, 2000.

[11] Roy Featherstone. *Robot Dynamics Algorithm.* Kluwer Academic Publishers, Norwell, MA, USA, 1987. Manufactured By-Kluwer Academic Publishers.

[12] Y. Fujimoto, S. Obata, and A. Kawamura. Robust biped walking with active interaction control between foot and ground. In *International Conference on Robotics and Automation (ICRA)*, pages 2030–2035. IEEE, 1998.

[13] Michael Girard and Anthony A. Maciejewski. Computational modeling for the computer animation of legged figures. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, volume 19, pages 263–270, July 1985.

[14] Michael Gleicher. Motion editing with spacetime constraints. In *1997 Symposium on Interactive 3D Graphics*, pages 139–148, April 1997.

[15] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522 531, August 2004.

[16] Radek Grzeszczuk and Demetri Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 95*, Annual Conference Series, pages 63–70, August 1995.

[17] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *International Conference on Robotics and Automation (ICRA)*, pages 1321–1326. IEEE, 1998.

[18] Jessica K. Hodgins and Nancy S. Pollard. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 153–162, August 1997.

[19] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, pages 71–78, August 1995.

[20] A. Hofmann, S. Massaquoi, M. Popovic, and H. Herr. A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1952–1959. IEEE/RSJ, 2004.

[21] O. Khatib. A unified approach to motion and force control of robot manipulators: the operational space formulation. *International Journal of Robotics Research*, 3(1):43–53, 1987.

[22] Oussama Khatib, Luis Sentis, Jae-Heung Park, and James Warren. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43, 2004.

[23] Oussama Khatib, James Warren, Vincent De Sapio, and Luis Sentis. *Human-Like Motion From Physiologically-Based Potential Energies*, volume XII of *On Advances in Robot Kinematics*, chapter Humanoids and Biomedical Applications. Springer, New York, 2004.

[24] Hyeong-Seok Ko and Norman I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2):50–59, 1996.

[25] Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. Planning motions with intentions. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 395–408, July 1994.

[26] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568,

August 2004. In Press.

[27] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[28] Paul G. Kry and Dinesh K. Pai. Interaction capture and synthesis. *ACM Transactions on Graphics*, 25(3):872–880, July 2006.

[29] Joseph F. Laszlo, Michiel van de Panne, and Eugene L. Fiume. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, pages 155–162, August 1996.

[30] Philip Lee, Susanna Wei, Jianmin Zhao, and Norman I. Badler. Strength guided motion. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, volume 24, pages 253–262, August 1990.

[31] A. Liégeois. Automatic supervisor control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(12):868–871, 1977.

[32] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416, July 2002.

[33] Per Lötstedt. Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. *Journal of Scientific Statistical Computing*, 5(2):370–393, 1984.

[34] A. A. Maciejewski. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications*, 10(3):63–71, 1990.

[35] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070, August 2005.

[36] Y. Nakamura and H. Hanafusa. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[37] Zoran Popović and Andrew P. Witkin. Physically based motion transformation. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, August 1999.

[38] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, Annual Conference Series, pages 349–358. ACM SIGGRAPH, July 1991.

[39] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[40] Charles F. Rose, Peter-Pike J. Sloan, and Michael F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3):239–250, 2001.

[41] Alla Safonova, Jessica Hodgins, and Nancy Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3):514–521, August 2004.

[42] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4):505–518, 2005.

[43] Hyun Joon Shin, Lucas Kovar, and Michael Gleicher. Physical touchup of human motions. In *Proceedings 11th Pacific Conference on Computer Graphics and Applications*, pages 194–203, 2003.

[44] Siddhartha Srinivasa. *Control synthesis for dynamic contact manipulation.* PhD thesis, Carnegie Mellon University, 2005.

[45] Adnan Sulejmanpasić and Jovan Popović. Adaptation of performed ballistic motion. *ACM Transactions on Graphics*, 24(1):165–179, January 2005.

[46] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, Annual Conference Series, pages 225–234. ACM SIGGRAPH, August 1990.

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

[48] Pierre-Brice Wieber and Christine Chevallereau. Online adaptation of reference trajectories for the control of walking systems. *Robotics and Autonomous Systems*, 54(7):559–566, July 2006.

[49] David A. Winter. *Biomechanics and Motor Control of Human Movement.* John Wiley and Sons, Inc., New York, 2nd edition, 1990.

[50] Andrew Witkin and Michael Kass. Spacetime constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, volume 22, pages 159 168, August 1988.

[51] Andrew Witkin and Zoran Popović. Motion warping. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, Annual Conference Series, pages 105 108. ACM SIGGRAPH, August 1995.

[52] Wayne Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans.* PhD thesis, Georgia Institute of Technology, 1998.

[53] Katsu Yamane, James J. Kuffner, and Jessica K. Hodgins. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics*, 23(3):532–539, August 2004.

[54] Katsu Yamane and Yoshihiko Nakamura. Dynamics filter—concept and implementation of online motion generator for human figures. *IEEE Transactions of Robotics and Automation*, 19(3):421–432, 2003.

[55] K. Yin, M. Cline, and D. K. Pai. Motion perturbation based on simple neuromotor control models. In *Pacific Conference on Computer Graphics and Applications (PG)*, pages 445–449, 2003.

[56] Victor B. Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *Symposium on Computer Animation (SCA)*, pages 89–96, July 2002.

[57] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701, August 2005.