Animating Virtual Characters using Physics-Based Simulation

Thomas Geijtenbeek

Cover design by Thomas Geijtenbeek

ISBN: 978-94-6182-389-2

Animating Virtual Characters using Physics-Based Simulation

Het Animeren van Virtuele Karakters met behulp van Fysica-Simulatie

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof. dr. G.J. van der Zwaan,
ingevolge het besluit van het college voor promoties in het openbaar
te verdedigen op dinsdag 17 december 2013 des ochtends te 10.30 uur

door

THOMAS GEIJTENBEEK

geboren op 2 oktober 1976
te Eindhoven

Promotor:	Prof.dr. R.C. Veltkamp
Co-promotor:	Dr.ir. A.F. van der Stappen

The work described in this thesis has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO), as well as the GALA project, funded by the European Union in FP7.

CONTENTS

1

INTRODUCTION

Believability is essential for virtual environments; any anomaly that can distort the acceptance of an alternate reality must be avoided. In cases when virtual environments attempt to imitate our world, a disrespect of our laws of physics is a common source of aberration. The incorporation of physical constraints offers an ongoing challenge for the animation of characters and objects that inhabit such a virtual environment.

Traditionally, the motion trajectories of virtual characters and objects are hand-crafted by skilled animators. This is a time-consuming process, and the resulting animations are purely kinematic: they have no regard for force or mass. The physical validity of such motion trajectories depends solely on the skill of the animator. As an alternative, people have been capturing performances from live actors. The resulting trajectories are guaranteed to describe natural motion, but the approach is limited to characters and motions for which an equivalent live performance is available. Furthermore, the naturalness of these motions may not be preserved during the processing that is often required to leverage limited motion data.

Animation becomes even more challenging during interaction; there are many ways in which virtual entities can interact with each other, and subtle variations may call for substantially different responses. For example, a stack of boxes may collapse in infinitely many ways, even with similar initial perturbations. Animation systems that draw from existing data (either captured or hand-crafted) require a complex process involving events, rules and geometric processing to generate proper responses. Despite great advances in data-driven methods in the last decade, their ability to produce plausible and non-repetitive responsive animation is restricted by the contents of the database.

Physics-based simulation offers a fundamentally different approach to

computer animation. Instead of directly manipulating the motion trajectories of objects and characters, this approach lets all motion be the result of a *physics-based simulation process*. As a consequence, physics-based characters and objects automatically interact in accordance with the laws physics, without the need for additional motion data or scripting.

Over the past decades, physics-based simulation has become an established method for the animation of passive phenomena, such as cloth, water and *rag-doll* characters. The conception that physics-based simulation can also be used to simulate actively controlled characters dates back to the early stages of 3D computer animation [WB85, AG85], and has incited many research papers since (see Figure 1.1). However, commercial animation frameworks still resort to kinematics-based approaches when it comes to animating active virtual characters [PP10, HZ11].

To understand this reservation, it is important to recognize the scope and complexity of controlling simulated characters. Just like with real-world entities, the pose of a physics-based character is controlled indirectly, through *forces* and *torques* generated by *actuators* that reside inside the body. As a result, the global position and orientation of a physics-based character cannot be controlled directly, but only through deliberate manipulation of external contacts. This impediment – also referred as *underactuation* – poses a direct challenge to basic tasks such as balance and locomotion. Such a challenge has no equivalent in traditional kinematics-based animation; it bears a much closer relation to humanoid robotics and control theory.

On the other hand, the primary focus of animation is *visual quality*. Several (especially early) physics-based controllers that emerged from robotics research focus on robustness, and have little attention paid to motion style. Consequently, many of these otherwise impressive results have been deemed stiff and robotic when compared to data-driven alternatives [NF02, WFH09, LKL10]. An important insight here is that even though these simulated characters move in a way that is *physically* valid, their motion is not necessarily *biomechanically* accurate [WHDK12]. The amount of torque that can be produced by real-world muscles is restricted by various physiological and mechanical properties, and biological control systems possess significant delays in neural processing and muscle activation. These *biomechanical constraints* are incorporated in any behavior we witness in nature, and play an important role in the perception of naturalness. However, the actuation models used in physics-based character animation are often highly simplified and do not enforce any of these constraints. As a result, such simulated characters often behave differently from biological equivalents, even if their motion is optimal within the constraints of the model [LHP05]. One possible way to overcome this is to incorporate captured motion data into the control strategy, but this limits motions to be similar to the data that is available [HZ11]. More recently, researchers have begun incorporating biomechanical

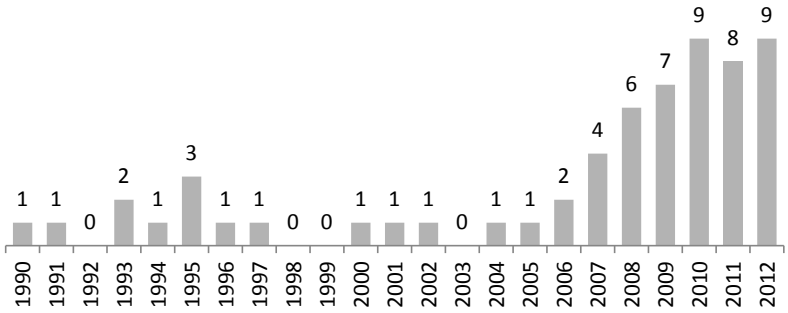


Figure 1.1: The yearly number of publications on physics-based character animation in *ACM*, *Eurographics* and *IEEE* journals and conference proceedings.

constraints into the character model, based on results from biomechanics research. This trend is gradually changing the direction of physics-based character animation research, and is further broadening its scope.

To implement all these aspects into a robust and flexible framework is a daunting task. Even though professional physics-based simulation software has become readily available, successful implementation of a physics-based character animation framework requires at least some knowledge of multi-body dynamics, numerical integration, control theory, biomechanical modeling and optimization theory. In addition, many physics-based control strategies require skillful and time-consuming manual tuning before they can be put to use – a process often poorly documented in research publications. Finally, physics-based character animation is computationally much more expensive than kinematics-based alternatives. It is only since about a decade that passive physics-based characters can be simulated in real-time on a consumer-grade PC. Currently, real-time performance of many state-of-the-art controllers is limited to a single character at a time on modern hardware.

In spite of these ramifications, recent trends have shown a renewed interest in using simulated physics for interactive character animation (see Figure 1.1). After decades of floundering, the field is maturing, with many recent publications demonstrating tremendous progress in both robustness and visual quality. As such, physics-based character animation remains an exciting research topic that is likely to play an increasingly important role in computer animation in the years to come.

1.1 Motivations and Goals

My fascination with physics-based character animation started after watching a video presentation of Karl Sims' monumental work on *evolving virtual creatures* [Sim94]. His work demonstrated various virtual creatures that had been evolved within a simulated environment. Those creatures were optimized for swimming, walking, jumping and fighting behaviors, and they exhibited both a personality and vulnerability unlike anything I had seen before. The work is a unique combination of several exciting research topics: evolutionary biology, motion control, artificial intelligence and computer graphics. Dedicating four years of research to these topics (and more) has been a joy, and the experience has brought me many insights and a much richer understanding of the science behind these topics.

Partly because of the exploratory nature of my PhD trajectory, the research topics and contributions in this thesis are quite diverse. The motivations and goals inspiring new research have been emerging as my knowledge of the subject increased. The remainder of this section is dedicated to describing these motivations and goals, for each of the aspects explored in this thesis.

An Overview of Related Work

One of the first observations during my PhD was the lack of consensus on how to classify the various methods described in research publications on physics-based character animation. For a novice researcher, this made the field rather difficult to dive into and become familiar with. From this observation evolved the idea to write a survey paper that would provide a structured evaluation of different aspects, approaches and techniques regarding physics-based character animation. The result of this work was accepted for a 90 minute *state-of-the-art report* presentation at *Eurographics* 2011 [GPEO11]. It has subsequently been edited into a publication in *Computer Graphics Forum* in 2012 [GP12], which forms the basis for Chapters 2 and 3.

Motion Evaluation using Musculoskeletal Models

Another early observation was the importance of evaluation of character animations. The assessment of perception of an animation is crucial when comparing new technology to current technology. However, research in this field is very limited; mainly because perception studies often require extensive user surveys – a task more related to psychology than computer science.

Instead of performing user studies, it is also possible to evaluate the physical accuracy of animations, based on the assumption that people are sensitive to physical inaccuracies in motion. However, existing research

publications that performed such an evaluation did not incorporate biomechanical actuation constraints, and therefore had limited accuracy. The goal of this research was to investigate if it would be possible to use existing models from biomechanics research to detect elements of motion that were biomechanically unrealistic, in order to work towards an automated evaluation method. The results of this research were presented at the 2010 *Motion in Games Conference* [GvdBvBE10], and form the basis for Chapter 4.

Injury Assessment

Even though several publications on physics-based character animation demonstrate impressive robustness during locomotion and balance tasks, at some point even the most robust controller will lose its balance and fall over. Without an appropriate falling strategy, physics-based characters can display highly erratic behavior in such circumstances, or simply drop dead like passive rag-doll characters. For use in a professional applications such a behavior is undesirable; characters that lose balance should fall in a convincing manner.

Out of this observation emerged the idea for 'falling controllers': motion controllers that attempt to break the fall of a character in a natural way. In order to be able to evaluate the effectiveness of such a controller (and possibly use an optimization scheme to find proper control parameters), it would be useful to have a measure of the injury of a character during fall. Coincidentally, such a measure could be used in gaming scenarios that require a more accurate evaluation of injury, even for passive rag-dolls. As such, we decided to dedicate a separate research project to the realistic assessment of injury for simulated characters. The results of this research were presented at the 2011 *Motion in Games Conference* [GVE11], and form the basis for Chapter 5.

A Simple Data-Driven Control Method

Implementability is crucial for mainstream acceptance of physics-based character animation methods. To be a likely candidate for early adoption, a method should be easy to integrate into existing physics-based simulation software; have high performance (several times faster than real-time); and have intuitive control for motion style. In existing research, there are several examples that are fast and easy to integrate, but for which the authoring of *motion style* involves cumbersome tuning or elusive high-level optimizations. The use of pre-recorded motion data can help in controlling style for physics-based characters; even though this limits controllers to data that is available, physics-based simulation still ensures physically valid handling of external perturbations and changes in character morphology. However, existing

methods that incorporate motion data required either manual preprocessing, or use on-line optimization methods based on the equations-of-motion of a character. Controllers based on on-line optimization methods are much harder to implement and integrate with existing physics-engines, and have relatively high performance requirements (real-time performance or slower).

Following this observation, it seemed a useful contribution to mainstream acceptance of physics-based character animation to devise a control method that can be driven using unprocessed captured motion data, has high performance, and can be easily integrated into existing engines. Additionally, such a method could be an important step towards controlling a physics-based character using live motion data – a scenario that has many possible applications. The result of this work has been presented at the 2012 *ACM/Eurographics Symposium on Computer Animation* [GPvdS12], and forms the basis for Chapter 6.

Natural *De Novo* Locomotion Controllers

In recent years, the incorporation of biomechanical constraints into physics-based control methods has proven an effective method to generate *de novo* (that is, without the use of preexisting motion data) natural-looking animations [GH10, WHDK12]. However, methods that focus on locomotion control had a number of limitations. First, existing methods were specifically catered towards realistic human gait. Therefore, these methods could not be used for the simulation of other (possibly imaginary) creatures or disproportional humans – both not uncommon in virtual environments. Second, the use of biomechanical constraints in these controllers was limited to the sagittal plane and lower body, meaning that sideways and upper-body motion were still not 'natural'.

Based on these observations, we have investigated the possibility to develop controllers in which all actuation and feedback is subject to biomechanical constraints, and in which the routing of the muscles is found through optimization, enabling our method to be used with creatures for which no musculoskeletal data exists. The result of this work has been accepted for presentation at the 2013 *SIGGRAPH Asia* conference, and will be published in a forthcoming issue of *ACM Transactions on Graphics*. It forms the basis for Chapter 7.

1.2 Thesis Outline

The remainder of this thesis is separated into three parts, each of which covers a different aspect of research on physics-based character animation.

Part I – Fundamentals

The first part of this thesis describes several fundamental aspects of physics-based character animation.

Chapter 2 contains a basic description of the modeling and simulation techniques that have been used in physics-based character animation. It is an introductory chapter, intended for readers that have little knowledge or experience with these topics.

Chapter 3 contains a structured overview of the various control strategies that have been used for physics-based characters. It is intended both as a thorough introduction for readers with an interest in physics-based character animation, as well as a convenient source of reference for researchers already familiar with the subject.

Part II – Physics-Based Motion Assessment

The second part of this thesis contains the results of research on how simulation and modeling techniques can be used to assess different aspects of motion, without the use of a control strategy.

Chapter 4 describes a method for evaluating the physical realism of kinematic animations using musculoskeletal models resulting from biomechanics research.

Chapter 5 describes a method for measuring the injury levels of a simulated character, using a model that relates results from research on human injury response to parameters in physics-based animation systems.

Part III – Physics-Based Character Animation

The third part of this thesis is based on the results from our research on actively controlled physics-based characters.

Chapter 6 describes a framework for controlling physics-based bipeds in a simulated environment, based on a variety of reference motions.

Chapter 7 describes a muscle-based control method for simulated bipeds in which both the muscle routing and control parameters are optimized. The resulting framework supports a wide variety of bipedal creatures, finds different gaits based on target speed, copes with uneven terrain and perturbations, and can steer to target directions.

Part I

Fundamentals

2

SIMULATION AND MODELING

All motion in physics-based character animation is the result of *physics simulation*, which is performed by a component commonly called a *physics simulator* or *physics engine*. Such a component iteratively updates the state of a virtual environment based on physics principles, resulting in animation of the individual elements that are part of the environment. Control of these elements is admissible only through application of external forces and torques (see Figure 2.1).

2.1 Constrained Rigid Body Simulation

In physics simulation, the modeling of material properties, contacts and friction can occur at many different levels of detail; there is generally a trade-off between accuracy and performance. In physics-based character animation, it is common to use what is referred to as *constrained rigid body simulation*, where *rigid* indicates that bodies are non-penetrable and

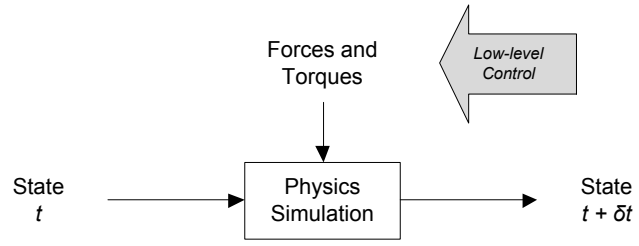


Figure 2.1: Animation using physics simulation. Control occurs only through application of forces and torques.

non-elastic, while their motion is *constrained* because body parts are linked together through *joints*. A physics simulator generally performs the following steps:

1. *Collision detection* investigates if intersections exist between the different object geometries.
2. *Forward dynamics simulation* computes the linear and angular acceleration of each simulated object.
3. *Numerical integration* updates positions, orientations and velocities of objects, based on the previously computed accelerations.

Step 2 and 3 are coupled in cases where *implicit integration methods* are used, *i.e.* when positions and velocities are formulated as functions of accelerations.

There are many textbooks available that describe methods for collision detection (*e.g.* [Cou01]), all of which are beyond the scope of this thesis. In this section we focus on the stages *after* collision detection. However, one topic that we will briefly mention is *self-collision*, *e.g.* collision between two legs of a biped character. Most publications on physics-based character animation ignore this type of collision and allow legs to penetrate each other. It is also common to ignore collisions between bodies that are directly connected via a joint. For such bodies, *joint limits* are imposed instead (see Section 2.2.1).

2.1.1 Newton-Euler Laws of Dynamics

The state of a single rigid body is described not only by *position* and *orientation*, but also by *linear velocity* and *angular velocity*. Change in linear velocity v depends on mass m, and the applied force F:

$$m\frac{\partial v}{\partial t} = F \tag{2.1}$$

Change in angular velocity ω depends on how the mass is distributed with respect to the center of mass. This is commonly represented by an *inertia tensor*, which is a 3×3 matrix that contains the moments of inertia about the principal axes of an object. Given inertia tensor I and applied torque T, the relation with angular velocity ω is:

$$I\frac{\partial \omega}{\partial t} + \omega \times I\omega = T \tag{2.2}$$

Forces and torques can be *external* (*e.g.* gravitational forces or collision response forces) or *internal* (*e.g.* the result of joint constraints or muscles contracting).

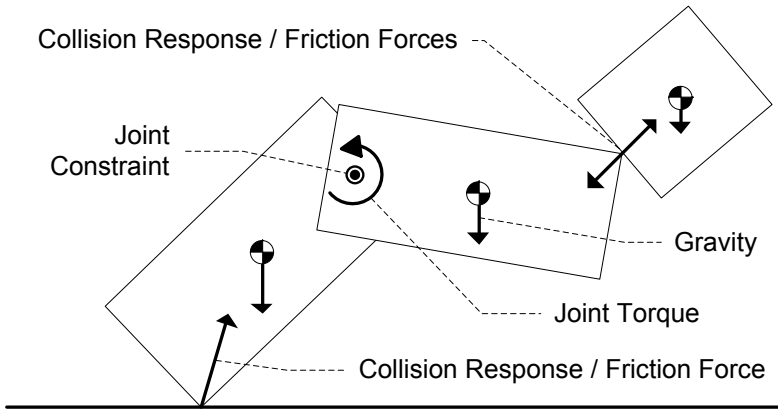


Figure 2.2: Different elements in forward dynamics simulation.

2.1.2 Joint Constraints

The different body parts of a physics-based character are held together through *joints*, which restrict the motion of the connected bodies (see also Figure 2.2). In forward dynamics simulation, there are two basic approaches to enforce these constraints. One approach is to apply *constraint forces* to the two bodies connected by the joint. These forces must always be of equal magnitude and in opposite direction.

Alternatively, the dynamics of a system of bodies can be described using an n-sized vector representing the *generalized* or *reduced* degrees of freedom (DOFs) of the system, $q = [q_1, \ldots, q_n]$:

$$\mathbf{M}(q)\ddot{q} + c(q, \dot{q}) + \mathbf{T}(q)\tau = 0 \tag{2.3}$$

in which $\mathbf{M}(q)$ is an $n \times n$ matrix representing mass distribution in generalized form (depending on pose q), $\ddot{q}$ are the accelerations of the generalized DOFs, $c(q, \dot{q})$ represents gravitational, centrifugal and Coriolis forces (which are virtual forces due to a changing reference frame). The n-sized vector τ represents internal actuation torques (see section 2.2.3), as well as forces and torques caused external contacts, while $\mathbf{T}(q)$ is an $n \times n$ coefficient matrix that has no specific meaning and depends on the form of $\mathbf{M}(q)$.

Using a generalized coordinates notation, joint constraints can be enforced by reducing the number of degrees of freedom in Equation (2.3) in such a way that they only represent unconstrained motion. Simulation using this approach is referred to as *reduced coordinate* simulation, while the former approach is referred to as *full* or *maximal coordinate* simulation. The use of reduced coordinate simulation omits the need for constraint

force tuning, but is more vulnerable to numerical instability in the case of near-singularities. Full coordinate simulation, on the other hand, has the benefit that constraint forces enable a simple mechanism for emulating flexible constraints that can be the result of ligaments and tissue.

Algorithms for constructing $\mathbf{M}(q)$, $\mathbf{T}(q)$ and $c(q, \dot{q})$ are described by Kane and Levinson [KL96]. Featherstone [Fea08] describes an efficient algorithm for performing reduced coordinate simulation.

2.1.3 Collision Response and Friction

There are two ways to respond to collisions: by applying a *penalty force*, or by constructing a *collision constraint*. A collision response force consists of two components: one that is in the normal direction of a collision surface and pushes objects away from each other to prevent penetration, and one that is parallel to a collision surface and is the result of *friction*. When contacts are not sliding, the magnitude of the friction force is limited by the force in the normal direction and the material properties. This relation is often modeled using a *Coulomb friction model*, which can be formulated as:

$$\|F_{xz}\| \leq \mu \|F_y\| \tag{2.4}$$

where F_y is the normal component of the collision force, F_{xz} is the parallel friction component of the collision force, and μ is a friction coefficient. The volume spanned by the set of possible reaction forces has the shape of a cone, and is often referred to as the *Coulomb friction cone*. Dynamic friction (sliding) will occur when a required collision response force lies outside this cone. To increase performance, physics simulators often approximate the friction cone using a pyramid with a 4-sided base [MLPP09, MTP12].

When collision response is modeled through constraints, a symbolic joint is constructed at the point of collision that prevents inter-penetration of colliding objects. This link is removed when objects are pulled away from each other, or transformed into a sliding constraint when the inequality in Equation (2.4) no longer holds.

Even though it seems like a small aspect of physics simulation, the way in which contacts and friction are modeled can have a significant impact on motion control and the performance of a physics-based character [MdLH10, JL11a].

2.1.4 Numerical Integration

After the accelerations of the virtual objects are known, they must be integrated to acquire updated velocity and position. A numerical integrator takes the position, velocity and acceleration at time t (q_t, $\dot{q}_t$ and $\ddot{q}_t$), and computes the updated position and velocity at time $t + \delta t$ ($q_{t+\delta t}$ and $\dot{q}_{t+\delta t}$).

The choice of integration technique and size of time step δt are crucial for the numeric stability and performance of the simulation. Typically, a larger time step will decrease simulation stability (caused by accumulation of integration errors), while a smaller time step will decrease simulation performance. Several methods have been developed to allow greater robustness at larger time steps, such as the *Runge-Kutta methods*. We refer to a textbook [WL97] for details on numerical integration methods for dynamics simulation.

2.1.5 Inverse and Mixed Dynamics

Instead of computing the accelerations for a known set of joint torques, it is also possible to do this the other way around: compute the torques and forces required for a character to perform a specific motion. This process is called *inverse dynamics*. It is often used in biomechanics to analyze the dynamics of human motion, based on motion data that is augmented with external force measurements [EMHvdB07]. Inverse dynamics is also used in physics-based character animation to find the torques required to achieve a desired acceleration [Isa87, FDCM97, YCP03, HMPH05, MZS09, LKL10]. It is also possible to combine forward and inverse dynamics, *e.g.* to control one half of a virtual character kinematically and the other half dynamically, by adding additional kinematic constraints to the dynamics equations. Such an approach is referred to as *mixed dynamics* [Ott03, vWvBE*10].

2.1.6 Available Physics Simulation Engines

There are several readily available software libraries that implement collision detection, forward dynamics, and numerical integration in a single package. Popular physics engines include the *Open Dynamics Engine* (ODE) [Smi06], *Bullet* [`bulletphysics.org`], *PhysX* [`nvidia.com`], *Havok* [`havok.com`], and *Newton* [`newtondynamics.com`]. ODE is by far the most commonly used third party simulator in research publications on physics-based character animation. Other publications use internally developed reduced coordinate simulation, or use packages that produce efficient code for computing the elements of the matrices involved in the *equations of motion* (see also Equation 2.3) for a specific character model. Examples of such packages are *SD / Fast* [HRS94] and *Autolev* [KL96]. Collision detection and response are not integrated in these packages. Another related software package is *Euphoria*, which focuses on controlling physics-based characters in real-time, using proprietary algorithms.

An in-depth comparison between different engines is beyond the scope of this thesis; however, the performance of various engines has been evaluated by Boeing and Bräunl [BB07], using basic (non-character) simulations,

as well as by Giovanni and Yang [GY11], using simulations of character locomotion.

2.2 Character Modeling

Characters in a simulated physics-based environment must incorporate a number of attributes that are not required for kinematics-based characters. They require mass properties and joint constraints, as well as *actuators* to enable motion control – each of which translates to elements in Equation (2.3). In this section we describe basic principles of physics-based character modeling, as well as different actuation models.

2.2.1 Structure

Most physics-based characters are modeled after humans, while some research papers model characters after animals [RH91, GT95, CKJ*11, TGTL11], robots [Hod91, RH91], or creatures that do not exist in nature [Sim94, TGTL11]. Character models are often simplified to increase simulation performance and to make them easier to control. Sometimes humans are reduced to simple biped characters, with head, arms and trunk modeled using a single body [YLvdP07, SKL07]. The feet of physics-based humanoid characters are also mostly modeled using single bodies, although Wang et al. [WFH09] include a separate segment for toes in their foot model to allow better locomotion performance. In later work, they add cylinders to the foot model to allow for foot rolling after heel-strike [WHDK12].

The choice of character model can sometimes be linked to the control strategy of the character. An illustrative example from robotics are the so-called *passive-dynamic walkers*, which are biped structures that walk downhill robustly without actuation [McG90]. Modified versions of these passive-dynamic walkers are able to perform biped locomotion on straight terrain, with very little actuation [TZS04, CRTW05].

Bodies The individual body segments of a physics-based character require both a *geometry*, which is used for collision detection, as well as *mass information*, which is used during forward dynamics simulation. Body segments are typically modeled using primitive shapes, such as boxes, spheres or cylinders, to increase collision detection performance. These shapes are also often used to compute the mass information of the individual bodies, assuming uniform mass distribution. Alternatively, mass properties can be derived from cadaveric data [ZSC90, dL96, VDJ99]. The collision detection geometry is independent from the geometry used for visualization, even

though coherence between the two is likely to increase perceived realism during collision response.

Joints The bodies of a physics-based character are connected through various types of joints. Commonly, mechanical joint types are selected to approximate natural constraints imposed by bone tissue and ligaments. For instance, knee and elbow joints are often modeled using hinge joints, while hip and shoulder joint are often modeled using ball-and-socket joints. Joints are constrained by *joint limits*, which are specified as lower and upper bounds for each DOF, and are used to mimic natural joint limits. The final range of motion of a character can be verified by comparing it to actual human motion data [Fro79]. Kwon and Hodgins [KH10] add sliding joints below the knee and the hip to emulate shock absorption due to soft tissues and ligaments. Full coordinate simulation systems (such as the Open Dynamics Engine [Smi06]) often provide a mechanism to tune joint constraints to emulate this behavior [GVE11].

2.2.2 Contact

Researchers sometimes explicitly model passive mechanisms for energy storage and release during contact. Raibert and Hodgins [RH91] use non-linear springs to model the padding material that some creatures have under their feet. Liu et al. [LHP05] model the springy tissue of a character's shoe sole. Pollard and Zordan [PZ05] use approximated soft contacts in simulated grasping motions. Jain and Liu [JL11a] show that soft contact modeling applied to the hands and feet of a character can increase robustness during locomotion, as well as allow greater control during object manipulation tasks.

2.2.3 Actuation

Characters need forces and torques to actively move around. In order for such forces or torques to be realistic, they must originate from within the character. We use the term *actuators* to describe the mechanisms that generate the forces and torques that make a character move.

Joint Torques The most straightforward actuation model is to generate joint torques directly for each actuated DOF. This way, the number of DOFs of the actuation model is the same as the number of active DOFs in the character. This actuation method can be visualized by assuming there exists a *servomotor* in each actuated joint, applying torques to its connected limbs.

Muscle Contraction Biological systems are actuated through contraction of *muscles*, which are attached to bones through *tendons*. When muscles contract, they generate torques at the joint(s) over which they operate (a single muscle can span multiple joints), causing the attached limbs to rotate in opposite directions. In addition to contracting when activated, muscles (and tendons, to a lesser degree) have the ability to stretch. This makes them behave like unilateral springs and allows for an efficient mechanism for energy storage and release [LHP05, AP93]. A state-of-the-art model for muscle dynamics is described in [GH10], and is used for animating the sagittal motion of a 3D virtual character in [WHDK12].

Since muscles can only pull, at least two muscles are required to control a single DOF (so-called *antagonistic muscles*). Biological systems often contain many more redundant muscles. In physics-based character animation, use of muscle-based actuation models is uncommon, because of the increased number of DOFs that require control and decreased simulation performance [WGF08]. However, examples of (often simplified) muscle-based actuation do exist [GT95, HMOA03, SKP08, LST09, MY11, WHDK12], and we witness an increased interest in using more advanced muscle-based actuation models for controlling physics-based characters [GH10, WHDK12].

External Forces In the real world, characters are controlled through forces or torques originating from inside a character. However, for animated characters this restriction does not apply; they can also be controlled through external forces and torques – similar to a puppet master controlling a puppet by pulling its strings. The direct control over global translation and rotation greatly simplifies the control strategy, but the fact that real-world characters do not possess such 'actuators' may result in loss of realism. For example, Wrotek et al. [WJM06] use external forces to give characters supernatural balance capabilities. To emphasize this aspect, external forces are also referred to as the *Hand-of-God* [vdPL95].

Actuation Limits In biological systems, actuation is limited by the maximum force of a muscle. In simulated environments too, actuation can be limited to prevent characters from appearing unnaturally stiff or strong, or to increase the stability of the simulation. With muscle-based actuation models, maximum forces can be derived from biomechanics research [vdH94]. However, with joint torque actuation models used in animation research, maximum torque values are generally fixed estimates per DOF [LWZB90, KB96, OM01], even though maximum torques in biological systems are dynamic and depend on many factors, such as the current length of a muscle, the contraction speed, the relative length of the contractile element, and the current activation state [GH10, vdBBH11].

3

MOTION CONTROL

Motion control is the central ingredient of any physics-based character animation framework. A *motion controller* is a component responsible for generating *actuation patterns* that make a physics-based character perform its goal tasks; it is an essential part of the character's *virtual brain*. Possible goal tasks include walking, running, picking up objects, stepping over obstacles, *etc.* – all while maintaining balance and withstanding unexpected external perturbations.

Similar to brains in biological systems, the computation of actuation patterns is based on *sensor data* that provides feedback from the state of the virtual environment and the character itself (see Figure 3.1). Such a control method is referred to as *closed-loop control*, denoting the presence of a closed feedback loop. Sensor data can represent a character's state

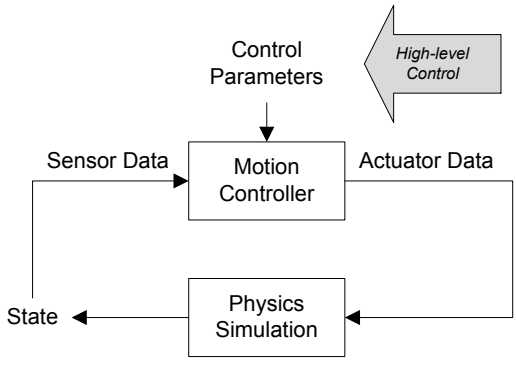


Figure 3.1: Closed-loop motion control.

(*e.g.* joint orientation, center of mass (COM) position and velocity, or total angular momentum [GK04]), it can represent external contact information (such as a *support polygon*), or it can be a task-related parameter, such as the relative position of a target object.

The opposite *open-loop control* (or *feed-forward control*) refers to a system that uses no sensor data or feedback loop. The use of open-loop control is uncommon in physics-based character animation, although feed-forward torques are sometimes used to complement torques from a closed-loop control system (see Section 3.1.3).

Motion controllers allow interaction with physics-based characters through *high-level control parameters*, such as desired walking speed, direction, or goal task. Because of this level of abstraction, physics-based characters typically possess a certain degree of autonomy, although some research papers focus on *direct user control* of physically simulated characters, through interface devices such as keyboard and mouse [LvdPF00, KP11, MPP11, WPP13], accelerometers [SH08], or motion capture [IWZL09, NWB*10].

Design Approaches Motion controller design is a multidisciplinary topic, with ties to several research areas, including biomechanics, robotics, control theory, artificial intelligence and optimization theory. There are many different angles from which to explore and classify attempts at motion control for physics-based characters, and there is currently no consensus as to a good classification scheme. Here, we classify different approaches based on the primary research field from which they originate:

- *Pose-driven feedback control* (Section 3.1). This approach originates from classic control theory; it attempts to control characters by defining kinematic targets for all actuated joints, and utilizes kinematics-based *feedback controllers* to compute the torques that minimize the difference between current pose and target pose.

- *Dynamics-based optimization control* (Section 3.2). This approach originates from optimal control theory and optimization theory. It attempts to find the actuator values (*i.e.* joint torques) that are optimal with regard to a set of high-level objectives, subject to the *equations of motion* describing the constrained dynamics of the character model and its environment.

- *Stimulus-response network control* (Section 3.3). This approach originates from artificial intelligence and is inspired by biological systems; it attempts to achieve motion control by using a generic control network (such as an *artificial neural network*) to construct explicit relations between input sensors and output actuators. Such control frameworks

typically contain no *a priori* knowledge of the control task or character; the set of optimal control parameters is sought through off-line optimization, according to high-level objectives.

Independent of the design approach, there are several techniques that are commonly used in motion control for physics-based characters. First of all, motion controllers often maintain a *simplified model* of a physics-based character to help simplify a control strategy. An example is the *Inverted Pendulum Model* (IPM), which is often employed to model the motion of a character's COM during single-stance phase, which in turn is used to predict an optimal target position for the swing leg [TLC*09, CBvdP10].

Also important are the provisions for controlling *motion style*, to reflect mood, personality or intention of a character. Some methods provide style control through editing key poses, others allow inclusion of continuous motion data. The use of recorded motion data allows easy authoring of specific motion styles, but controllers that depend on motion capture data suffer from some of the same limitations as kinematics-based data-driven animation systems.

Motion controllers may benefit from off-line optimization of their internal parameters, based on high-level goals such as *walking speed* or *energy efficiency* – analogous to learning or evolving in biological systems. However, finding an appropriate high-level optimization metric is difficult, because the parameters that describe natural motion are considered elusive and enigmatic [vdP00], while simulated physics-based characters have different optimums than biological equivalents, because they often ignore important physical attributes [LHP05, WHDK12].

To achieve higher-level composite behaviors, individual motion controllers can be combined in a meta-control framework. For this purpose, Faloutsos et al. [FvdPT01, Fal01] have developed a meta-control framework that allows flexible transition from one controller to another, independent of controller type and implementation. Physics-based controllers may also be used in combination with kinematics-based controllers to get the best of both worlds. For more details on this topic, we refer to the works of Shapiro et al. [SPF03] and Zordan et al. [ZMCF05, ZMM*07].

3.1 Pose-Driven Feedback Control

This approach attempts to control physics-based characters by providing *kinematic target trajectories*, which are tracked using *feedback controllers* that attempt to minimize the difference between the measured state and target state. It has its origin in *control theory* and is a common control method in industrial robotics [KSnL05].

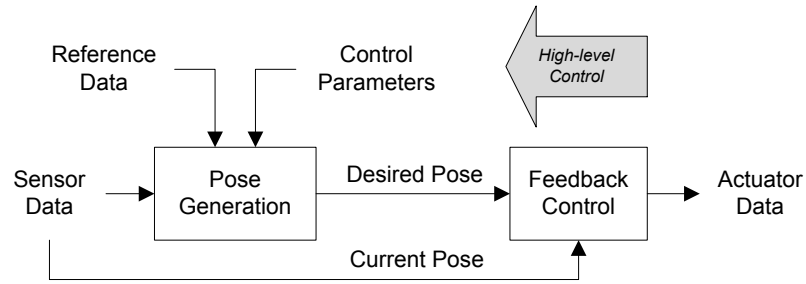


Figure 3.2: Pose-Driven Feedback Control

Figure 3.2 shows a schematic overview of this control method for physics-based character animation. Its main component performs *balance and pose control*, based on the current state, high-level control parameters, and (optionally) a time-based reference pose. It outputs a *desired kinematic state*, usually in the form of desired orientation and angular velocity for each joint. After that, a set of *feedback controllers* compute joint torques based on the difference between measured and target state.

The main appeal of this approach is its ease of implementation: all motion can be specified in the kinematic domain, based on kinematic insights, without the need for direct access to the underlying equations of motion. The downside is that kinematics-based feedback controllers have no knowledge of the dynamics of the system (*i.e.* the character), which makes the effect of the resulting joint torques often unpredictable and non-optimal. Feedback controllers also require elaborate tuning to adapt to characters with different dynamic properties, although this can effectively be automated using off-line optimization techniques [HP97, YCBvdP08, WFH09, WFH10].

In the remainder of this section, we first describe techniques for feedback control, followed by an overview of different approaches for generating kinematic targets.

3.1.1 Proportional-Derivate Control

The most widely used feedback control technique that uses kinematic targets is called *proportional-derivative control*, or *PD control*. It computes a joint torque, τ, that is linearly proportional to the difference between the current state and a desired state. Physics-based characters generally contain rotational joints only; kinematic targets are therefore mostly represented through joint orientation and angular velocity (however, feedback control strategies can be applied similarly to prismatic (*i.e.* sliding) joints).

A PD torque is based on both the difference between current orientation θ and desired joint orientation θ_d, as well as the difference between current angular velocity $\dot{\theta}$ and desired angular velocity $\dot{\theta}_d$:

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

In this equation, $\mathrm{k_p}$ and $\mathrm{k_d}$ are *controller gains*, and they regulate how responsive the controller is to deviations in position and velocity, respectively.

Often, applications only specify a desired joint orientation and set desired angular velocity to zero ($\dot{\theta}_d = 0$). A PD controller then becomes similar to a spring-damper system, where a spring generates a force to move to its rest position, θ_d. In such a system $\mathrm{k_p}$ defines the spring gain (or *spring constant*), and $\mathrm{k_d}$ the damping.

Parameter Tuning Finding correct values for $\mathrm{k_p}$ and $\mathrm{k_d}$ is not a straightforward task, and often requires manual tuning through *trial-and-error*. When controller gains are set too low, a joint may not be able to track its target and lag behind. When set too high, a joint may follow a target trajectory too rigidly and become stiff and unresponsive. The amount of rigidity is also referred to as *impedance* – in skilled human motion, impedance is usually low [TSR01, Hog90], while high impedance motion is regarded as robotic or stiff [NF02].

The gains $\mathrm{k_p}$ and $\mathrm{k_d}$ must also be set in a proper relation to each other. A relatively high value for $\mathrm{k_p}$ may result in *overshoot* (meaning that a joint will oscillate around its desired position before reaching it), while a relatively high value of $\mathrm{k_d}$ may cause unnecessary slow convergence. When the relation between $\mathrm{k_p}$ and k_d is optimal, a controller is said to be *critically damped* (meaning convergence is fastest, without overshoot). If the desired state and dynamics characteristics are fixed, the optimal relation between the two gain parameters is: $\mathrm{k_d} = 2\sqrt{\mathrm{k_p}}$. For physics-based characters this relation is more complex, as both the desired state and the dynamics characteristics are subject to constant change. Still, this relation is often used as an estimate or a starting point for tuning.

To decrease the amount of tuning, Zordan and Hodgins [ZH02] scale controller gains in accordance to an estimate of the moment of inertia of the chain of bodies controlled by a joint. They also increase responsiveness by temporarily lowering controller gains for significant body parts when a perturbation is detected. Another way to decrease manual tuning is to use off-line optimization of the gain parameters [vdP96, HP97].

Stable PD Control Tan et al. [TLT11] present a modification to the standard PD control formulation that increases stability at high gains. Instead of using the current position and velocity to compute joint torques, they

estimate the position and velocity during the upcoming time step, based on the current velocity and acceleration. These estimates are then used in Equation (3.1), together with the desired position and velocity during the upcoming time step.

PID Control

When affected by external forces such as gravity, the steady state of a PD controller may be different from the desired position [NF02]. Industrial control systems often add an *integral* component to the equation that compensates for accumulated error between current and target position. Such a control method is called *proportional-integral-derivative control*, or *PID control*. However, in character animation such an approach has very limited use, because any change in desired state invalidates the current integral error.

Antagonist Feedback Control

Inspired by muscle-based actuation, Neff and Fiume [NF02] propose *antagonist feedback control* as an alternative to PD control. For each actuated DOF, a pair of antagonistic springs operate in opposing direction to create a joint torque and simultaneously regulate impedance. Each spring has a fixed set point, located at either joint limit. Instead of setting a target angle, this method varies the spring gains to achieve a desired position. Since both springs have linear control, there exists a mathematical equivalence between PD control and antagonist control. The main advantage of antagonist control is that its parameters allow for more intuitive tension control.

3.1.2 Jacobian Transpose Control

The essence of Jacobian Transpose Control is that it enables control in Cartesian space for linked structures with redundant DOFs. Using the Jacobian Transpose, it is possible to compute the set of torques that emulate the effect of an external force or torque, applied to a specific body or a virtual location, such as the center-of-mass. As a result, it allows feedback control in task space instead of joint space.

Jacobian Transpose Control is a common control strategy in robotics [SADM94], and an important part of the *virtual model control* strategy of Pratt et al. [PCTD01]. In physics-based character animation, Jacobian Transpose Control has been used for controlling end-effectors, balance, gravity compensation and target velocity [MZW99, CBvdP10, CKJ*11, GPvdS12], usually on top of per-joint PD control.

Virtual forces and torques are applied to a *chain of linked bodies*, starting from a static base link (such as the stance foot) and moving to a target link. The set of joint torques τ_F that emulate a virtual force F applied at point p corresponds to:

$$\tau_F = J(p)^T \; F \tag{3.2}$$

where $J(p)$ is the *Jacobian* that represents the rate of change of a point p for each DOF i connecting the targeted chain of bodies. For a chain of bodies connected through k rotational DOFs, each row in $J(p)^T$ represents the rate of change of p with rotation α_i about DOF i:

$$J(p)^T = \begin{bmatrix} \frac{\partial p_x}{\partial \alpha_1} & \frac{\partial p_y}{\partial \alpha_1} & \frac{\partial p_z}{\partial \alpha_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial p_x}{\partial \alpha_k} & \frac{\partial p_y}{\partial \alpha_k} & \frac{\partial p_z}{\partial \alpha_k} \end{bmatrix} \tag{3.3}$$

For a rotational DOF i represented by normalized axis a_i and anchor position b_i (all defined in the world coordinate frame), the derivative $\frac{\partial p}{\partial \alpha_i}$ corresponds to the cross product between a_i and the relative position of p [Jaz10]. Hence, each row i of $J(p)^T$ corresponds to:

$$J(p)_i^T = \begin{bmatrix} \frac{\partial p_x}{\partial \alpha_i} & \frac{\partial p_y}{\partial \alpha_i} & \frac{\partial p_z}{\partial \alpha_i} \end{bmatrix} = (a_i \times (p - b_i))^T \tag{3.4}$$

A *virtual force* F applied at point p can now be emulated by applying a torque $\tau_{F,i}$ to each DOF i that is part of the chain of bodies:

$$\tau_{F,i} = (a_i \times (p - b_i))^T \; F \tag{3.5}$$

Similarly, it is possible to apply a virtual torque to a specific body at the end of a chain. If the orientation of the targeted body is represented using an *exponential map* [Gra98], $e \in \mathbb{R}^3$, then the rate of change of e with rotation α_i is identical to the normalized DOF axis a_i:

$$J(e)_i^T = \begin{bmatrix} \frac{\partial e_x}{\partial \alpha_i} & \frac{\partial e_y}{\partial \alpha_i} & \frac{\partial e_z}{\partial \alpha_i} \end{bmatrix} = a_i^T \tag{3.6}$$

Hence, a *virtual torque* T applied to the body at the end of the chain can be emulated by applying a torque $\tau_{T,i}$ to each DOF i that is part of the chain of bodies:

$$\tau_{T,i} = a_i^T \; T \tag{3.7}$$

Note that, since virtual characters contain no links that are truly static, the effect of a virtual force or torque is always an approximation and can be determined only after simulation [CKJ*11, GPvdS12].

3.1.3 Feed-Forward Control

A number of control methods use predefined feed-forward torque patterns (*i.e.* torque patterns that are constructed without sensory feedback information) on top of feedback torques that are based on kinematic targets. The motivation for this approach comes from evidence suggesting that biological systems use feed-forward control for most of their motions and use feedback control only for low-gain corrections [TSR01]. In physics-based animation, feed-forward torques have been acquired through optimization [GT95], inverse dynamics [YCP03], *feedback error learning* (FEL) [YLvdP07], and by using an on-line parallel auxiliary simulation of unperturbed motion capture data with high-gain tracking [NVCNZ08].

A problem with feed-forward torques is that they do not work well with discontinuous motions, because the precise timing of sudden torque change is not predictable in advance. An example is the sudden increase in torque during heel strike. As a result, feed-forward methods work best for continuous motions [GT95, YCP03], or for body parts that move more or less continuously during discontinuous motion, such as the upper body during walking [YLvdP07]. Exceptions exists though, as Yin et al. [YLvdP07] apply FEL to all joints when using the SIMBICON framework in combination with adapted motion capture data.

3.1.4 Procedural Pose Generation

Early attempts [RH91, HWBO95] generate motion using hand-crafted functions or procedures that attempt to model behaviors witnessed in biological systems. Such algorithms are developed based on human intuition, using insights from physics, robotics and biomechanics.

An early example and excellent illustration of this approach is the work of Raibert and Hodgins [RH91], who construct various gait types for a number of low-dimensional robot characters. Their character models include passive springs to emulate natural mechanisms for energy storage-and-release, enabling hopping style motions without explicit control. Control is governed by a *finite state machine* that tracks the current phase of the gait cycle, based on foot sensor data. During each phase, a set of control strategies work in parallel to control balance, speed and posture. One control strategy is responsible for target foot placement during swing phase and uses a simplified inverted pendulum model in combination with inverse kinematics. The foot placement not only handles balance, but also enables high-level speed control: the difference between desired speed and current speed is used to offset the foot placement. Another control strategy attempts to keep the upper body upright in world coordinates, by applying a torque to the hip and knee joints of the stance leg.

Extending this approach, Hodgins et al. [HWBO95] develop controllers that allow running, cycling and vaulting motions for a full-body human character. In their running controller, just before heel-strike, they adjust the hip torque to ensure the swing foot has zero relative speed when compared to the ground surface – a natural behavior called *ground speed matching*. Wooten and Hodgins [WH00] demonstrate controllers for leaping, tumbling, landing and balancing. Faloutsos et al. [FvdPT01, Fal01] demonstrate controllers for sitting, falling, rolling over and getting back up.

The development of procedural pose-driven feedback controllers is a skillful process that requires laborious trial-and-error, while the resulting controllers are often inflexible to even minor changes in character morphology or environment. Hodgins and Pollard [HP97] show how to adapt their running and cycling controllers to different character morphologies, but their method requires off-line optimization. Another issue with these controllers is that they focus on function instead of style. As a result, animations generated using these controllers appear robotic [NF02].

In recent research, Tan et al. [TGTL11] use procedural motion generation to animate underwater creatures, using target trajectories based on geometric functions. The parameters for these functions are found through off-line optimization for high-level criteria, such as maximization of swimming speed and minimization of energy consumption.

3.1.5 Pose Generation using Key Poses

Pose-control graphs are an attempt at a more generic approach for defining kinematic targets for physics-based characters. In this approach, each state in a finite state machine (represented by a directed graph) is linked to a *key target pose*, which is fed as input to local feedback controllers. The approach bears some resemblance to kinematic *keyframe animation*.

An advantage of the pose-control graphs method is that it allows intuitive creation or modification of different motion styles, without the need for detailed understanding of the underlying control strategy. Animators can create or tune controllers by simply editing the key poses. However, these key poses are not guaranteed to be reached during simulation; sometimes key poses need to be exaggerated, far beyond the intended pose, to get a desired effect. Key poses also need to be tuned in conjunction with the gain parameters of the local feedback controllers.

This basic approach has been used successfully for animating low-dimensional 2D characters [vdPKF94], but 3D biped control requires additional balance correction. The main reason for this is that there is no tracking of global translation and rotation of the target poses. Laszlo et al. [LvdPF96] use pose-control graphs for balanced 3D locomotion controllers using *limit-cycle control*. To maintain balance, they apply linear pose corrections at the be-

ginning of each walking cycle, based on pelvis rotation or center of mass. In later research, pose-control graphs have been used to generate a number of interactive skills such as climbing stairs [LvdPF00] or user-controlled skiing and snowboarding [ZvdP05]. Yang et al. [YLS04] developed a layered pose-base swimming controller.

SIMBICON Yin et al. [YLvdP07] introduce a generic framework for biped locomotion control, based on pose-control graphs. Their SIMBICON framework (short for SIMple BIped CONtroller) allows biped locomotion control with a large variety of gaits and styles. Example controllers can walk in different directions and perform various gaits such as running, skipping and hopping, using only two to four states. At the core of this control framework exists an efficient balance control strategy, which corrects foot placement using the swing hip and provides posture control using the stance hip.

The SIMBICON framework has been at the basis for a range of research projects. Coros et al. [CBYvdP08] use off-line optimization to develop a top-level control policy that switches between controller instances, based on the current state of the character and a goal task. Using this strategy, they demonstrate a controller that can walk over gaps while planning several steps ahead [CBYvdP08], as well as a controller that performs tasks with long-term position and heading goals, with increased robustness during perturbations and steering behaviors [CBvdP09].

Other researchers focus on off-line optimization of the parameters of the SIMBICON framework. Yin et al. [YCBvdP08] demonstrate how controllers can be optimized to step over obstacles, walk on ice, traverse slopes and stairs, and push or pull large crates. Wang et al. [WFH09] optimize the SIMBICON parameters to influence motion style, using a compound optimization metric that includes terms for energy minimization, gait symmetry and head stabilization. In subsequent research, Wang et al. [WFH10] increase robustness by optimizing controllers in simulations with added uncertainty and constrained environments. Both results use Covariance Matrix Adaption (CMA) [Han06] for optimization.

Coros et al. [CBvdP10] have extended the SIMBICON framework in a number of ways, resulting in a control framework for walking that no longer needs to be tuned for specific size, motion style or walking speed. Most importantly, they use *virtual forces* (see Section 3.1.2) to compensate for gravity, to control speed and for subtle balance control. The use of virtual forces defeats many of the issues caused by the uncoordinated nature of local feedback controllers. Another addition is the use of an inverted pendulum model to control swing foot placement, effectively making balance control independent from body height (similar to Tsai et al. [TLC*09]). Thirdly, they use key target poses to generate continuous spline-based target trajectories,

which increases robustness of tracking. Finally, they use *inverse kinematics* (IK) to adjust the upper body target position, allowing object manipulation tasks during locomotion. Even though the framework allows for various robust walking behaviors, it is not suitable for high energy locomotion such as running. In later research, Coros et al. [CKJ*11] apply similar techniques to develop a control framework for several four-legged creatures, incorporating a flexible spine control model.

3.1.6 Data-Driven Pose Generation

Motion capture trajectories seem an ideal source for kinematic targets for physics-based characters, since they are known to be physically feasible and include both target positions and target velocities (after proper filtering). However, there are some discrepancies between the original motion and the simulated motion, because of at least three reasons:

- A simulated physics-based character rarely has the exact same proportions and mass properties as the performing actor.
- Motion capture systems do not capture many subtle balance correction behaviors of the performing actor.
- In real-time physics engines, the simulation of real-world phenomena such as friction, tissue deformation or joint constraints are highly simplified at best.

Because physics-based characters are underactuated, errors in global translation and orientation can accumulate, eventually leading to loss of balance. Sok et al. [SKL07] attempt to resolve these discrepancies by 'fixing' the kinematic target trajectories, using off-line optimization of non-linear displacement maps. Their method increases performance of 2D biped locomotion tracking, but still requires an additional balance compensation strategy.

Zordan and Hodgins [ZH02] track full-body boxing and table tennis motions and combine it with an in-place procedural balance strategy, similar to that of Wooten and Hodgins [WH00]. In addition, they use inverse kinematics to adjust upper body motion trajectories, creating interactive boxing and tennis controllers. Yin et al. [YLvdP07] use the SIMBICON balance strategy for 3D locomotion, based on preprocessed motion capture data.

Lee et al. [LKL10] demonstrate a framework that can track various styles of 3D locomotion data, which can be delivered on-the-fly without the need for preprocessing. They achieve balance by modulating the reference data using SIMBICON-like balance strategies for stance hip, swing hip, stance

ankle and swing foot. For feedback control, they use inverse dynamics to compute joint torques from target accelerations, based on the current state and reference data. Strictly speaking, their method is different from other methods described in this section, because the use of inverse dynamics requires access to the *equations of motion*, which is not supported by many common physics engines.

State-Action Mapping This control strategy is based on the idea that the appropriate kinematic target of a physics-based character can directly be derived from the current pose of the character. The method maintains a set of mappings between current state and target pose, where a pose at time t (the source state) is mapped to a pose at time $t + \delta t$ (the target). The method is ideal for use with motion capture data, because both source and target can directly be derived from subsequent frames of motion data.

An early example of state-action mapping is *Banked Stimulus Response* [NM93], which has been used to develop several interesting 2D and 3D locomotion controllers (albeit without the use of motion capture data) [AFP*95]. Sharon and Van de Panne [SvdP05] use state-action maps that are initialized using motion capture data and optimized off-line. The active mapping is selected based on the current pose, using a nearest neighbor criterion. The resulting controllers demonstrate stylized 2D biped locomotion. Sok et al. [SKL07] extend this approach, using a much denser set of mappings, which are acquired from optimized motion capture clips of similar motions. Their input state includes velocity, foot position and ground contact information, while their target poses are augmented with joint velocities. Even though their framework allows for a wide range of motions and transitions between motions, it is still limited to 2D.

3.1.7 Summary

The use of feedback controllers based on kinematic targets is the most well-studied approach for controlling physics-based characters; this has resulted in a vast number of controllers for various types of balance, locomotion and interaction. The main strengths are its intuitiveness and ease of implementation.Control strategies can largely be defined in the kinematic domain and kinematics-based feedback controllers are easy to comprehend. Also, several recent publications have shown great robustness against external perturbations (*e.g.* [YLvdP07, WFH10]), while PD Control has shown to be an effective mechanism for simulating natural joint compliance.

A weakness of using kinematics-based feedback controllers requires laborious tuning or lengthy optimization, and that the lack of knowledge of the dynamics of the system leads to non-optimal control. As a result, early examples of pose-driven feedback controllers often appear stiff and

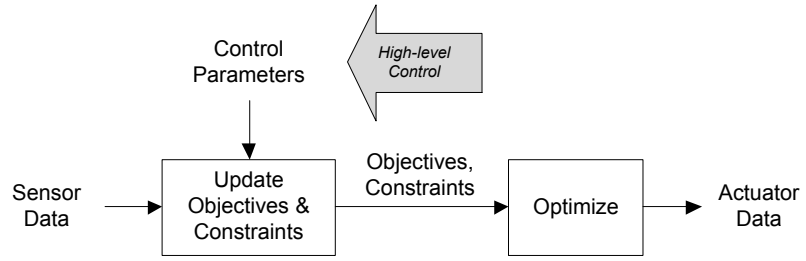


Figure 3.3: Dynamics-based optimization control.

robotic. The current state-of-the-art overcomes many of the problems of local feedback control by adding control techniques that allow for coordination amongst different actuators. Examples of this are the work of Coros et al. [CBvdP10, CKJ*11], who incorporate virtual forces in their control framework, and the work of Lee et al. [LKL10], who use inverse dynamics for tracking reference motion trajectories. Additionally, the use of feed-forward control (*e.g.* [YLvdP07]) implicitly embodies coordinated action. In parallel, research in off-line optimization of control parameters has resulted in impressive controllers, both for new behaviors [YCBvdP08], stylized locomotion [WFH09, WFH10], and swimming behaviors [TGTL11].

Both the implementations of the SIMBICON framework [YLvdP07] and the framework of Coros et al. [CBvdP10] are available on-line. In addition, DANCE [SFNTH05, SCAF07] is an open-source framework for developing motion controllers, with a focus on pose-driven control methods.

3.2 Dynamics-Based Optimization Control

We use the term *dynamics-based optimization control* to describe methods that compute *on-line* the set of actuator values that are optimal for a given goal. The dynamics of a character and its environment are formulated as a set of *constraints*, while the intended behavior of a character is defined through high-level *objectives*. The optimal set of actuator values is then acquired through *constrained optimization*. Figure 3.3 displays a schematic overview of this method.

There is a difference between *dynamics-based optimization control* and optimization methods described in previous sections: earlier methods attempt to find *control parameters* through *off-line* trial-based optimization, whereas the methods described in this section attempt to find *actuator values* (*i.e.* joint torques) through constant *on-line* optimization based on the equations of motion describing the character dynamics.

The main advantage of *dynamics-based optimization control* is that this approach explicitly incorporates the equations-of-motion in the optimization process, establishing a tight link between control and dynamics simulation. Compared to pose-driven feedback control methods, this ensures a much better prediction of the effect of applying a set of joint torques. A downside is that these control methods are more difficult to implement than previously described approaches; they require substantial knowledge of both multibody dynamics and constrained optimization – topics with which computer animation specialists may not be familiar. In addition, it is more difficult to model natural joint compliance (in the case of unexpected perturbations) using full-body torque optimization, when compared to pose-driven feedback control methods. Finally, these methods are computationally more expensive than kinematics-based feedback control equivalents, typically limiting real-time performance to a single character at a time.

This approach is related to *optimal control theory*, a method that has been applied to computer animation earlier by Witkin and Kass [WK88]. In their *spacetime optimization* framework, a user defines a set of constraints (*e.g.* a character must have a specific pose at a specific time), and a set of objectives (*e.g.* a character must use as little energy as possible), after which dynamically accurate motion trajectories are generated that meet these constraints and objectives. Based on this approach, several researchers have demonstrated physically realistic animations for different characters and behaviors [PW99, FP03, SHP04, LHP05, LYvdP*10]. Related to this is the work of Grzeszczuk et al. [GTH98], who use neural networks to learn the dynamics constraints from a physics-based character model.

Spacetime optimization is not suitable for interactive applications, because it optimizes motion sequences *as a whole* [LHP06]. Any unexpected disturbance invalidates the remaining trajectory; even numerical integration errors can be considered disturbances [LvdPF96, dSAP08a]. It is computationally very expensive to re-optimize the entire motion sequence at every time step. In Section 3.2.2 we provide an overview of strategies developed to overcome this challenge.

3.2.1 Constraints and Objectives

The basic task of any optimization-based motion controller is to solve the following optimization problem:

$$\underset{\tau_t}{\text{argmin}} \ \{G_1, G_2, ..., G_n\} \ , \ \text{subject to} \ \{C_1, C_2, ..., C_m\} \tag{3.8}$$

where G_i are the n high-level *objectives* and C_i describe the m *constraints*. The result is a set of joint torques, τ_t, that is optimal for the given goals and constraints at time t. This optimization is performed at regular intervals,

but typically at a lower rate than the update rate of the physics simulation itself.

Constraints The conditions that all parameters must fulfill during optimization are captured in constraints. The most important set of constraints ensure that solutions are in accordance with the Newton-Euler laws of dynamics. These constraints are represented in Equation (2.3), which describes the relation between a set of torques and forces τ (which include the set of joint torques τ_t), and a set of generalized accelerations $\ddot{q}$. Because of its form, it is called an *equality constraint*. An example of an *inequality constraint* is the Coulomb friction of Equation (2.4). Inequality constraints are also used to enforce bounds or limits, such as *joint limits* or *maximum torques* (see Section 2.2.1).

Objectives The behavior of the controller is shaped through objectives, usually through *cost functions* that need to be minimized. An example cost function that can be used for static balance measures the horizontal displacement between the character's COM and its center of support. Other example objectives minimize energy usage, or promote tracking of a reference motion. Objectives are also referred to as *soft constraints* [SC92b].

Note that on-line cost functions are different from cost functions (or fitness functions) used in off-line optimization. The key difference is that off-line optimization shapes the entire motion, while on-line optimization methods only involve the current time step. For example, a cost function based on maximum COM height would in *off-line optimization* promote jumping, where in *on-line optimization* it might promote tiptoeing and raising hands.

Multiple objectives can be combined into a single objective function using a weighted average. The downside of such an approach is that, apart from the additional tuning that is required, different competing objectives may interfere with each other. For instance, a motion tracking objective may interfere with a balance objective, causing a character to fall [AdSP07]. To get around this, de Lasa et al. [dLH09] demonstrate a solver that allows for prioritized optimization, by only meeting lower-priority objectives when they do not interfere with higher-priority objectives such as balance. Their method is based on *task-space* and *operation-space* formulations found in robotics research (*e.g.* [Lie77, Kha87]), which have also been used in physics-based animation research by Abe and Popović [AP06].

3.2.2 Control Strategies

The main challenge in dynamics-based optimization control is that the optimization must be performed on-line. We distinguish between three basic

approaches to overcome this challenge:

- *Immediate Optimization*. Only optimize for the current situation, without looking ahead. This approach works well for motions that do not require planning, such as static balance, or for motions constructed through finite state machines, similar to those described in Sections 3.1.4 and 3.1.5.

- *Preoptimized Prediction Models*. Use off-line optimization to find trajectories that are used to guide on-line optimization. This approach incorporates optimized look-ahead control information of a specific motion, without having to perform full trajectory optimization on-line.

- *Model Predictive Control*. Constantly reoptimize future motion trajectories, reflecting the current state. To manage computational costs, these methods optimize only for a short period ahead in time and often use simplified character models.

Each of these approaches will be described in detail below.

Immediate Optimization

The most straightforward approach to optimization-based motion control is to use a control strategy based on objectives that can be optimized for without looking ahead. Pioneering examples of this approach can be found in the work of Stewart and Cremer [SC92b, SC92a], who define state-driven constraints to construct controllers for bipedal walking, as well as climbing and descending stairs. Later examples include the work of Abe et al. [AdSP07], who use a weighted objective that drives the projected COM position towards the center of the base-of-support, while tracking a reference pose or motion. Optimal joint torques are found through *quadratic programming*. The result is an in-place balance controller that robustly copes with change in friction or character morphology, supporting uneven foot placement and moving support surfaces. Macchietto et al. [MZS09] demonstrate natural-looking and robust balance controllers that use objectives for angular momentum control. They use inverse dynamics for tracking the optimized target trajectories. Wu and Zordan [WZ10] use a similar approach, but add trajectory objectives for COM and swing foot to produce stepping behaviors for balance correction. Their trajectories are based on motion capture data and analysis of the character's momentum.

Other methods use finite state machines to construct more complex behaviors. Jain et al. [JYL09] demonstrate controllers for side-stepping, object dodging and balancing through use of external support from walls, railings or other characters. They use a weighted set of objectives, including

objectives that drive hands and feet towards specific locations. De Lasa et al. [dLMH10] construct robust balancing, walking and jumping controllers by combining several coordinated and state-driven low-level objectives, using prioritized optimization. They demonstrate how to tune objectives to suggest different moods and exhibit properties of natural human motion. In addition, their controllers are robust against major on-the-fly changes of character morphology.

Finally, da Silva et al. [dSDP09] demonstrate a framework in which different controllers with immediate optimization objectives can be combined, by weighting the contribution of different controllers based on current state and goal.

Preoptimized Prediction Models

Another strategy is the use of a *preoptimized prediction model*, which is a term we use to describe a model that incorporates motion trajectories that are computed through off-line constrained optimization, and which are adapted by an on-line controller based on the current state.

Da Silva et al. [dSAP08a] use a reference motion to optimize a three-link biped model for balanced locomotion. They then use a linear feedback policy to combine the model predictions with a tracking objective to generate real-time locomotion tracking. Muico et al. [MLPP09] use a reference motion to optimize a model of both a full-body character and ground reaction forces, and use a non-linear feedback policy for on-line tracking. Their controller can track agile walking and running motions with sharp turns. In later research [MPP11], they increase robustness against perturbations and inclines by simultaneously tracking multiple trajectories and using a graph that describes possible blends and transitions between trajectories. Wu and Popović [WP10] use off-line optimization to compute optimal end-effector trajectories and tracking gains, which are selected by an on-line controller at the beginning of each step, based on the current state and a goal task. On-line joint torques are found through *quadratic programming*. Their controller demonstrates robust locomotion on uneven terrain using sharp turns and backwards walking, without requiring reference motion trajectories, and can be used with common physics engines.

Model Predictive Control

This control strategy maintains a low-dimensional model of the character, which is used in on-line optimization to find a set of motion trajectories covering a short period ahead in time. These predictive target trajectories are constantly updated and used as input to control the high-dimensional character. The use of predictive motion trajectories enables real-time look-

ahead capabilities, which are important for actions involving complex root motions, such as jumping and rolling, but also for discontinuous dynamic behaviors, such as walking and running.

Model predictive control is related to humanoid robotics research that performs *preview control* of the *zero-moment point* (ZMP) of a character model [KKK*03b, KKK*03a, KKI06, WC06, Che07]. The ZMP is defined as the point from which a character's ground reaction force would result in a zero net moment [VB04]. A character is statically balanced if its ZMP coincides with its center-of-pressure (COP) [Wie02].

Da Silva et al. [dSAP08b] demonstrate a motion capture tracking controller that uses a linearized model of linked rigid bodies and contact forces for predictive control. Kwon and Hodgins [KH10] demonstrate a controller for running motions, using an inverted pendulum model optimized using motion data to generate reference footstep coordinates on-line. Mordatch et al. [MdLH10] demonstrate robust locomotion on highly constrained and uneven terrains, using a spring-loaded inverted pendulum model (an inverted pendulum model that uses a spring to model variable length of the stance leg) for on-line planning. Their resulting COM and foot trajectories are then tracked using low-level feature objectives described by De Lasa et al. [dLMH10].

Ye and Liu [YL10] demonstrate a controller that can track a reference motion with on-line modification of long-term goals and completion time. They use off-line optimization to construct an abstract model that describes a nonlinear relation between global motion and external contact forces. Their controller robustly responds to external perturbations and changes in step height, as long as the footstep positions in the response strategy do not deviate from the original motion. Jain and Liu [JL11b] demonstrate a model that allows on-line long-horizon planning at every time step. Their model is based on modal analysis of reference motion data and is used to construct separate control strategies for dynamically decoupled modes (an approach explored earlier by Kry et al. [KRFC09]). To improve performance, they use a linearized dynamics system and disregard higher-frequency modes. Their method allows robust tracking and on-line editing of a reference trajectory, but the linearized dynamics system prevents the control method to be used for high energy motions such as running.

3.2.3 Summary

The recent renewed interest in dynamics-based optimization techniques to control physics-based characters has led to several impressive results. Some of these results have not been demonstrated using pose-driven feedback control methods, including tracking of captured running motions, balance control on moving surfaces, and navigation over uneven terrain. The main

challenge of these methods is in the fact that optimization must be performed in real-time, which makes full trajectory planning infeasible. We have described three approaches to overcome this hurdle:

1. Describe optimization goals in such a way that they don't require looking ahead. This approach has led to various robust balance controllers [AdSP07, MZS09, WZ10], as well as a locomotion and jumping control framework that uses prioritized optimization [dLMH10].

2. Perform off-line trajectory optimization and use the results to guide the on-line optimization process. This has resulted in robust motion tracking controllers [MLPP09, MPP11] as well as flexible navigation over uneven terrain [WP10].

3. Use a simplified model to perform look-ahead trajectory optimization on-line. This approach has resulted in controllers for agile running [KH10], locomotion on highly constrained and uneven terrains [MdLH10] and robust motion tracking [YL10, JL11b].

Implementation The implementation of an optimization-based motion control framework is significantly more challenging than pose-driven feedback control methods. It requires substantial knowledge of constrained dynamics modeling and on-line optimization techniques such as quadratic programming – a skill set uncommon for developers with a background in computer animation.

The amount of tuning or preprocessing required for new characters or behaviors varies per method. Some methods require preprocessing for different motions or character morphologies (*e.g.* [MLPP09, MPP11]), while methods that combine low-level objectives to achieve high-level behaviors require tuning to get the behavior right (*e.g.* [JYL09, dLMH10]). On the other hand, some controllers are robust against large changes in character morphology, without the need for additional tuning (*e.g.* [AdSP07, dLMH10]).

The computational requirements for optimization-based controllers depend on the number of objectives, the degree in which they compete [JYL09], the complexity of the character model, and the number of active external contacts [MLPP09]. In general, computational requirements for optimization-based methods are substantially higher than most pose-driven feedback control methods; publications typically report real-time or near real-time performance for a single character.

3.3 Stimulus-Response Networks

Inspired by motion control found in biological systems, this approach attempts to control characters through a network that connects sensor data

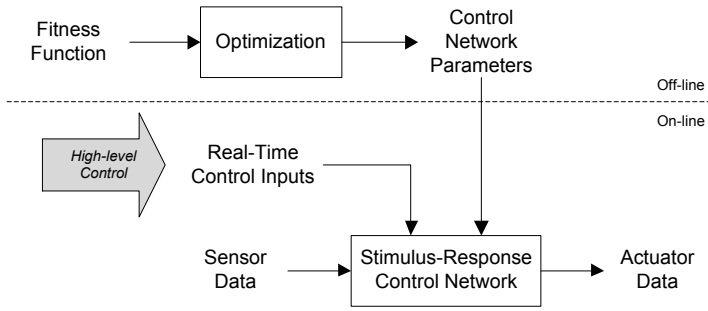


Figure 3.4: Stimulus-response network control using off-line optimization.

(the stimuli) to actuator data (the response). The parameters of such control network are found through optimization using a high-level optimization metric, often referred to as *fitness function*.

The main appeal of this approach is that it allows motion controllers to be constructed automatically using only an optimization metric; it requires no *a priori* knowledge of motion or character. However, devising such a metric is a daunting task that involves expert trial-and-error tuning. In addition, there seems to exist a trade-off between what a control network is capable of and its optimization performance. Control networks that are highly flexible often optimize poorly.

Even though optimization can be performed both off-line and on-line, research on physics-based character animation focuses on off-line optimization, where controller candidates are evaluated through short simulation runs during which the parameters do not change. The final optimized controller is the one with the highest fitness after a (usually large) number of trials. See Figure 3.4 for a schematic overview. On-line optimization occurs mostly in robotics [TZS04, MAEC07, BT08], where control parameters are adapted on-the-fly by a process called *reinforcement learning* [KLM96]. For robotics research, a generate-and-test approach is often impractical because of the large number of trials and damage risks, a notable exception being the work of Pollack et al. [PLF*00].

3.3.1 Stimulus-Response Network Types

Even though stimulus-response networks exist in many flavors, they all consist of interconnected processing elements (often called *nodes* or *neurons*), each with a variable number of inputs and a single output. Control networks are typically *recurrent*, which means their internal links form a loop. This allows for the representation of an internal state, or memory. Delays can be added to each node output to promote real-time dynamical behaviors inside

the control framework [vdPF93]. Nodes that are connected to sensor input are referred to as *sensor nodes*, while nodes that are connected to actuators are called *actuator nodes*.

Artificial Neural Networks The most common type of stimulus-response network is the *artificial neural network* (ANN), in which each node outputs a value based on the weighted sum of its input nodes and a *threshold function* [Bis94]. Van de Panne et al. [vdPF93] use such a network for controlling various low-DOF 2D characters. Reil and Husbands [RH02] demonstrate locomotion control for a low-DOF 3D biped, using a small circular ANN with fixed topology and no input sensors. Their outputs generate target joint angles, which are converted into joint torques using PD controllers. Allen and Faloutsos [AF09] use a network with a flexible topology that is optimized along with its parameters, using the NEAT method [Sta04]. Their approach has produced a set of amusing gaits, but no stable 3D biped locomotion control.

Genetic Programs Other stimulus-response networks are more related to *genetic programs*, in the sense that their processing nodes can perform logical operations (*and*, *or*, etc.), decision operations (*if*, *then*, *else*), or memory storage [Gar90]. In addition, the networks of Sims [Sim94] use several node types that generate periodic signals, such as sine or sawtooth waves.

Cyclic Pattern Generators An important related framework used in locomotion control is the *Cyclic Pattern Generator* (CPG) [Tag95]. The patterns produced by a CPG can represent joint torques, muscle activation, or target joint angles. CPGs have been used in combination with off-line optimization to produce biped locomotion controllers [TYS91, Tag95, Tag98, MTK98]. Grzeszczuk et al. [GT95] optimize CPGs that generate feed-forward muscle activation patterns, resulting in various swimming behaviors. The circular neural networks used by Reil and Husbands [RH02] are designed to emulate CPGs.

3.3.2 Fitness Function Design

Fitness functions used in physics-based motion control are often a weighted sum of individual terms with specific goals. Both formulation and weighting of the individual terms require manual tuning, often through time consuming trial-and-error. An example fitness function for developing locomotion controllers can be based on the following principles:

- To promote horizontal movement, increase fitness based on the horizontal distance of the COM from the origin after a fixed simulation time.
- To penalize falling down, decrease fitness when the vertical COM position falls below a certain threshold.

Using such basic approach, several researchers have succeeded in producing stable locomotion behaviors for 2D characters [vdPF93], 3D quadrupeds [AFP*95], and 3D bipeds [RM01, RH02]. Sims [Sim94] demonstrates several walking, running swimming and jumping behaviors for creatures with evolving morphology, using basic fitness functions. We have not seen any reports of stable locomotion controllers for full-body 3D humanoids (*i.e.* characters with arms, legs and a head) that have been developed using stimulus-response networks.

Several fitness functions have been designed for interactive behaviors. For example, the distance to a target object has been used to promote following or evading behaviors [Gar90, vdPF93, Sim94, RH02] or stepping over obstacles [Tag98]. Sims [Sim94] demonstrates creatures competing over the control of an object, evaluating fitness in head-to-head competitions. Grzeszczuk and Terzopolous [GT95] optimize a meta-control framework to generate high-level interactive behaviors from a set of low-level controllers.

There is little work on stimulus-response control that explicitly deals with motion style, or only at a very low level. Van de Panne [vdP00] states that "determining a proper optimization metric which captures the natural qualities of human and animal motions still remains enigmatic and elusive." Auslander et al. [AFP*95] demonstrate only crude style control by adding fitness terms based on average COM-height and step length. We have not found any reports that use stimulus-response network control with motion capture data as part of a fitness function. However, motion capture data has been used in methods based on *state-action maps* [SvdP05, SKL07], which can be regarded as simplified stimulus-response networks (see also Section 3.1.6).

3.3.3 Optimization Strategies

Even though off-line optimization can be applied to any parametrized control framework, stimulus-response network control is the only method that fully depends on it. This is not only part of the evolution-inspired philosophy behind this control strategy; it is simply the only perceivable way to set the parameters of a stimulus-response network in a meaningful way.

Optimization performance is linked to the shape of the *fitness landscape*, which is a visualization of fitness as a function of the parameters that are being optimized. Roughly speaking, good fitness landscapes have smooth

gradients and few local minima, and show little dependencies between individual parameters [RH02].

Papers that use stimulus-response network control for physics-based characters generally use *evolutionary algorithms* (EAs) for optimization. The choice of using EAs is often based on a design philosophy that states that behaviors that arise from such strategies are more natural [Ale01]. However, there is little evidence to support this claim; no research on stimulus-response network control compares different optimization methods. It is questionable whether EAs are efficient for optimizing motion controllers, as they heavily rely on computationally expensive fitness evaluation. Recent publications on physics-based character animation express a preference for using Covariance Matrix Adaption (CMA) [Han06] for off-line parameter optimization [WFH09, WFH10, WP10, TGTL11]. However, none of these control frameworks are based on stimulus-response networks.

Optimization Performance Apart from the choice of the type of control network and optimization technique, there are some additional techniques that can help increase optimization performance:

- *Initialization.* Results from previous optimizations may be used for initialization in subsequent optimizations.
- *Early Termination.* If a simulation is not promising in early stages, it can be terminated early in order to avoid wasteful computation [RH02].
- *Reward Shaping.* To smooth the fitness landscape, fitness functions can be designed in a way that minimizes local optima in the fitness landscape [NHR99, SvdP05].
- *Bootstrapping.* In addition to shaping the reward function, the fitness landscape can also be smoothed by gradually increasing the complexity of a task during the optimization [Gar90, SvdP05].

Increasing Robustness Auslander et al. [AFP*95] demonstrate that robustness of a controller can be improved by using random variation in the initial conditions during controller optimization. Wang et al. [WFH10] demonstrate how including noise or random perturbation during optimization can lead to more robust controllers. However, both these techniques have not been applied to stimulus-response network control strategies. Van de Panne et al. [vdPF93] note that stimulus-response controllers become less robust when they are optimized for more specific conditions.

3.3.4 Summary

In spite of promises based on early successes, recent achievements on motion control using stimulus-response networks are limited. Where off-line parameter optimization has shown to be an effective technique in other frameworks [CBYvdP08, YCBvdP08, WFH09, WFH10, TGTL11], we have seen no demonstration of full-body humanoid biped locomotion using stimulus-response networks. Apparently, the stimulus-response networks used for motion control are either too restrictive to allow for complex character control, or too flexible to allow for effective parameter optimization. The most distinctive quality of stimulus-response network control may very well be its capability to produce unexpected and entertaining animations (*e.g.* [Sim94, AF09]).

Part II

Physics-Based Motion Assessment

4

Evaluating Animations using Musculoskeletal Models

Perceived realism of movement is an important component of character animation. However, its subjective nature makes it an impractical quality. Therefore, it is desirable to develop automatic evaluation methods that approximate the perceived realism of motion.

There are several ways to construct such an evaluation method. One approach is through analysis of results from user studies. Another approach is to develop a measure based on statistical analysis of existing realistic motions. In this chapter, we present a method that is based on the idea that there exists a relation between the *perceived realism* and the *physical realism* of a motion; the main rationale being that all motion we witness in real life adheres to the laws of physics and biomechanics.

Our goal is not to test this hypothesis, but to develop a method that provides an unprecedented level of accuracy, which allows for detailed user studies on the difference between perceived and physical realism. Eventually, this may lead to a validated automatic perception measure.

Current methods that measure physical realism are all limited in their ability to describe human motion, since they do not consider the fact that the joint torques that produce motion are the direct result of muscle forces acting on these joints. The amount of torque a muscle can generate depends on both its maximum force as well as its muscle moment arm, which is pose-dependent. In addition, several muscles operate over multiple joints, creating complex dependencies between joint torques. These aspects can only be accurately described through a model of the human musculoskeletal system.

In the field of computer animation, the development of such muscu-

loskeletal models may be considered too specialized and labor-intensive to be worth the effort. However, in biomechanics research, realistic musculoskeletal models are considered an effective tool for conducting research on human motion. As a result, significant effort has been put into the development of such models. However, in order to allow exact computation of all joint moments, motion data must be augmented with force measurements at external contact points. We have developed a method for estimating these external force measurements, thus enabling the use of musculoskeletal models directly with purely kinematic motion data. In addition, we show how a musculoskeletal model can be employed to measure two principally different types of physical realism.

Our first measure evaluates to what degree a character motion obeys the *Newton-Euler laws of motion*, which dictate that changes in linear and angular momentum must be consistent with external forces due to gravity and contact. Examples of animations not obeying these laws are a character hanging still in mid-air, or a character standing straight at a 45 degree angle without falling. Our measure reflects these errors in a way that uniformly applies to both flight and contact stages.

Our second measure evaluates the realism of the muscle force that a human character would require to perform a motion. This measure detects situations where for example a character is moving too fast, or jumping too high. Such motions need not be in conflict with the Newton-Euler laws of motion, but are physically unrealistic because they require an excessive amount of muscle force.

Both quality measures provide objective feedback on the physical realism of an animation, with a relatively high level of accuracy.

4.1 Related Work

Human character animation has received a lot of attention during the past decades. There are several classes of techniques, each having its own advantages and disadvantages. See Van Welbergen et al. [vWvBE*10] for an overview of different animation techniques.

The amount of research conducted on the perception of physical realism of animation is limited. O'Sullivan et al. [ODG03] define a number of measures regarding perception of physical realism of the motions of solid objects, based on user studies. Reitsma and Pollard [Rei03] observe through user studies that errors in horizontal velocity and added accelerations in human jumping motions are easier observed by humans than errors in vertical velocity and added decelerations. Point light experiments have shown that observers can accurately estimate lifted weight [RF81], as well as pulled weight [MV98] from point light displays.

Much work has been done on physical realism in generated motions. Safonova and Hodgins [SH05] show that certain basic physical properties can be conserved during motion interpolation. Ikemoto et al. [IAF07] evaluate transitions by foot skating and by evaluating the zero-moment point [VJ69]. Ren et al. [RPE05] present a tool to evaluate the realism of animations using a set of statistical models, based on natural example data. Some work makes use of user studies to evaluate the resulting animations. For example, Van Basten and Egges [vBE09] evaluate the relationship between posture distance metrics and perceived realism of motion transitions. Some techniques modify existing animations to improve the physical realism. For example, Shin et al. [SKG03] show how motions can be adjusted in such a way that they satisfy the zero-moment point constraint (*i.e.* ensure that the center of pressure is located within the support polygon of a character), as well as some additional physical constraints. Ko and Badler [KB96] attempt to achieve physical realism in generated walking motions by constraining maximum joint torque and adjusting balance in an inverse dynamics analysis.

Erdemir et al. [EMHvdB07] provide an overview of the different approaches and applications of musculoskeletal modeling. Veeger and Van der Helm [VvdH07] provide an exemplary insight in the complexity of musculoskeletal mobility. In recent years, a number of tools have implemented these techniques. Examples of such tools are the commercially available *AnyBody* system [RDS*03], the *Human Body Model* [vdBGEZ*13], and the open source project *OpenSim* [DAA*07]. To date, we are unaware of researchers that have used musculoskeletal models to evaluate computer animations.

4.2 Assessment Method

In this section we will describe in detail how a musculoskeletal model can be used to evaluate the physical realism of a character animation. We will describe how to compute measures representing the error in the *dynamics* of the animation, as well as the error in the amount of *muscle force* a human would require to perform an animation. Our method is independent of the musculoskeletal model or modeling software that is being used; we will therefore omit implementation specific aspects of musculoskeletal modeling.

4.2.1 Prerequisites

Musculoskeletal Model

Our evaluation method uses a model that incorporates both the skeletal and the muscular structure of the entire human body. Such a full-body musculoskeletal model can formally be described as a tuple:

$$\{\mathbf{B}, \mathbf{Q}, \mathbf{U}\} \tag{4.1}$$

in which set **B** describes the individual body segments of the model. Each segment has mass and inertial properties, as well as a reference position and orientation. The set **Q** describes the n kinematic degrees of freedom of the model (including global position and orientation). The set **U** describes the l muscles in the model. For each muscle, it describes the relation between joint angles and muscle moment arm, as well the maximum force the muscle can produce. The exact form of **B**, **Q** and **U** depends on the implementation of the model and is not relevant to our evaluation method.

Animation

The input for our method is an animation **A**, consisting of a sequence of T frames:

$$\mathbf{A} = \{A_1, A_2, ..., A_T\} \quad , \quad A_t = \{q_t, \dot{q}_t, \ddot{q}_t, S_t^l, S_t^r\} \tag{4.2}$$

where q_t is a pose vector representing the n kinematic degrees of freedom defined in **Q**; $\dot{q}_t$ and $\ddot{q}_t$ are first and second order derivatives of q_t. S_t^l and S_t^r are polygons representing the ground contact region at frame t, for both left and right foot. These support polygons can be derived from q_t, and depend on the geometry of both the model and the ground. We do not regard this procedure as part of our method, and assume the support polygons are included with the animation.

4.2.2 Quality Measures

When measuring the realism of an animation, there are two aspects to consider. First, a realistic animation must satisfy the laws of dynamics (*i.e.* all motion must be the result of permitted forces and torques); second, an animation must respect the strength limits of a character. Our method captures these elements in the following quality measures:

1. The *dynamics error* measure, $\kappa : \mathbf{A} \rightarrow \mathbb{R}^+$, which is defined as the total amount of combined external force and torque that is required for a motion to satisfy the Newton-Euler laws of motion. In other words, it describes the amount of 'magical support' required to make a motion dynamically valid. The measure uniformly applies to both contact and flight stages. Force and moment are normalized using body mass and height, enabling comparison between character models of different weight or height.

2. The *muscle error* measure, $\lambda : \mathbf{A} \rightarrow \mathbb{R}^+$, which is defined as the total amount of excess muscle force required for a character to perform **A**, normalized by the total maximum force of all muscles in **U**. The excess muscle force is defined as the amount of force a muscle must

produce on top of its maximum capacity. The normalization enables comparison between character models of different strength.

4.2.3 Method Details

Both quality measures are calculated per frame, for each $A_t \in \mathbf{A}$. This is done in four consecutive steps, as depicted in Figure 4.1.

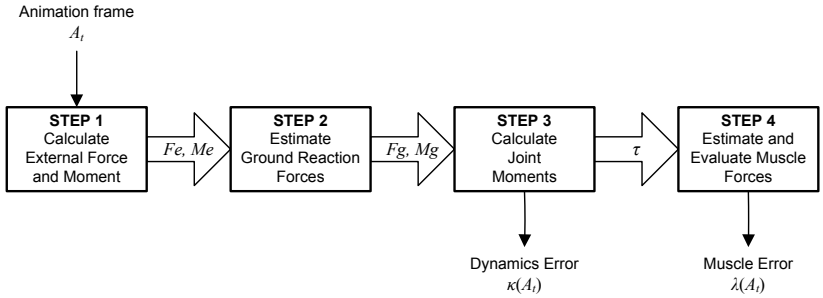


Figure 4.1: Overview of the consecutive steps that are performed for each frame A_t.

Step 1: Calculate External Force and Moment

In this step we compute the external force and moment F_e and M_e that act on the character at frame A_t. We do so by solving the dynamics equations of the motion defined by q, $\dot{q}$ and $\ddot{q}$, which can be formulated as:

$$\mathbf{M}(q)\ddot{q} + c(q,\dot{q}) + \mathbf{T}(q)\tau = 0 \tag{4.3}$$

where $\mathbf{M}(q)$ is an $n \times n$ matrix that describes the pose-dependent mass distribution, based on **B** and **Q**. The vector $c(q,\dot{q})$ encodes gravitational, centrifugal and Coriolis forces. The vector τ contains the (unknown) moments and forces acting on the degrees of freedom in **Q**. $\mathbf{T}(q)$ is an $n \times n$ coefficient matrix whose form depends on $\mathbf{M}(q)$). The approach to constructing $\mathbf{M}(q)$, $\mathbf{T}(q)$ and $c(q,\dot{q})$ is not relevant for our method and depends on the modeling software that is employed (for example, see [KL96]). The only important aspect for our method is that global position and orientation are part of the dynamics equation. After re-arranging the components in (4.3) we get:

$$\tau = -\mathbf{T}(q)^{-1}\left[\mathbf{M}(q)\ddot{q} + c(q,\dot{q})\right] \tag{4.4}$$

After solving, the external force and moment acting on the root element of the character correspond to the elements in τ related to global position

and rotation, as defined in $\mathbf{Q}$. If F_r and M_r are the force and moment defined in the root coordinate frame, with the origin at r and orientation $\mathbf{R}$, then the external force and moment in the global coordinate frame, F_e and M_e, are defined as:

$$F_e = \mathbf{R}^{-1}F_r \quad , \quad M_e = \mathbf{R}^{-1}M_r + r \times (\mathbf{R}^{-1}F_r) \tag{4.5}$$

Step 2: Estimate Ground Reaction Forces

In this step, we estimate the ground reaction force and moment for both left foot, F_g^l and M_g^l, and right foot, F_g^r and M_g^r. The estimate is based on external force and moment F_e and M_e, support polygons S_t^l and S_t^r, and a static friction coefficient μ. We use a right-handed coordinate system in which the positive y-axis is pointing upwards, and we assume that the ground plane is defined by $y = 0$. The ground reaction forces must meet the following constraints:

1. At least one foot must be in contact with the ground before there can be any ground reaction force ($S_t^l \cup S_t^r \neq \emptyset$).

2. The external force F_e must point upwards, otherwise it cannot be attributed to ground contact: $F_{e,y} > 0$ (with $F_{e,y}$ being the vertical component of F_e).

3. The horizontal component of the ground reaction force must be in agreement with the Coulomb friction model: $\|F_{e,xz}\| \leq \mu \|F_{g,y}\|$ (with $F_{e,xz}$ being the horizontal component of F_e).

4. The origin of the ground reaction force and moment for each foot must lie within the support polygon of the corresponding foot.

In our method, we do not employ a dynamic friction model. All ground reaction forces are considered the result of static friction.

If constraint 1 and 2 are not met, no external forces or moments will be applied to the character during step 3 ($F_g^l = M_g^l = F_g^r = M_g^r = 0$). Otherwise, we first compute the total ground reaction force and moment, F_g and M_g, and distribute these among both feet afterwards.

We assume that the upward component of F_e can be fully attributed to ground reaction force: $F_{g,y} = F_{e,y}$. We do not limit the maximum amount of ground reaction force in the upward direction, since this is only limited by the amount of pushing force a character can produce. If such a pushing force is not realistic, we will see this reflected in the muscle error $\lambda(A_t)$.

To meet constraint 3, we limit the magnitude of the horizontal component of the ground reaction force, $F_{g,xz}$, based on friction constant μ and the vertical ground reaction force $F_{g,y}$:

$$F_{g,xz} = \begin{cases} F_{e,xz} & \text{if } \|F_{e,xz}\| \leq \mu \|F_{g,y}\| \\ \frac{\mu \|F_{g,y}\|}{\|F_{e,xz}\|} F_{e,xz} & \text{if } \|F_{e,xz}\| > \mu \|F_{g,y}\| \end{cases} \tag{4.6}$$

The ground reaction moment around y, $M_{g,y}$, is also limited by contact friction between the character and the ground plane. However, since we do not expect large moments around y, we assume that any such moment is automatically countered by static friction: $M_{g,y} = M_{e,y}$.

The ground reaction force and moment applied to each foot must originate from a point on $y = 0$ that lies within the respective support polygon. This imposes a constraint on the ground reaction moment around x and z. To apply this constraint, we first define c_e as the point on $y = 0$ from which F_e and M_e would originate:

$$c_{e,x} = \frac{M_{e,z}}{F_{g,y}} \quad , \quad c_{e,z} = \frac{-M_{e,x}}{F_{g,y}} \tag{4.7}$$

If c_e lies outside a support polygon S, we define the origin of the ground reaction force, c_g, as the point inside S that is closest to c_e; otherwise: $c_g = c_e$. The ground reaction moments $M_{g,x}$ and $M_{g,z}$ then become:

$$M_{g,x} = -c_{g,z} F_{g,y} \quad , \quad M_{g,z} = c_{g,x} F_{g,y} \tag{4.8}$$

In cases where only one foot is in contact with the ground we assign F_g and M_g to that foot. In cases where both feet are in contact with the ground, the computation of (4.8) is performed individually for each foot. The forces and moments are then weighted according to the distance between c_e and support polygons S_t^l and S_t^r. If $d : c \times S \rightarrow \mathbb{R}$ is the distance between a point c and support polygon S, then the weighting factors ω_l and ω_r become:

$$w_l = \frac{d(c_e, S_t^l)}{d(c_e, S_t^l) + d(c_e, S_t^r)} \quad , \quad w_r = \frac{d(c_e, S_t^r)}{d(c_e, S_t^l) + d(c_e, S_t^r)} \tag{4.9}$$

where w_l is the scaling factor applied to acquire F_g^l and M_g^l, and w_r is the scaling factor applied to acquire F_g^r and M_g^r.

Step 3: Calculate Joint Moments

Now that we have estimated ground reaction force and moment for both feet, we can add them to the dynamics equation:

$$\tau = -\mathbf{T}(q)^{-1} \Big[\mathbf{M}(q)\ddot{q} + \mathbf{E}(q) \Big[\begin{array}{cccc} F_g^{l\,T} & M_g^{l\,T} & F_g^{r\,T} & M_g^{r\,T} \end{array} \Big]^T + c(q, \dot{q}) \Big] \tag{4.10}$$

where $\mathbf{E}(q)$ is a $12 \times n$ coefficient matrix of which the form is dictated by $\mathbf{M}(q)$ (see [KL96]). After solving[1] for τ, F_r and M_r correspond to the elements in τ related to global position and orientation of the root element, in the coordinate frame of the root. They are the remaining external force and moment that could not be attributed to ground contact. Their magnitudes are used for the calculation of the dynamics error during frame A_t:

$$\kappa(A_t) = \frac{\|F_r(A_t)\|}{mg} + \frac{\|M_r(A_t)\|}{mgh} \tag{4.11}$$

where m is the subject body mass, g is the gravitational constant and h is the subject height. We employ these variables to ensure that both elements are dimensionless quantities, independent of weight and height.

Step 4: Estimate and Evaluate Muscle Forces

Each moment τ_i that is an element of moment vector τ is a result of the muscles acting on the degree of freedom q_i (except for those related to global position and orientation). The relation between the joint moments in τ and the vector of muscle forces, u, can be described as:

$$\tau = \mathbf{D}(q_t)u \tag{4.12}$$

where $\mathbf{D}(q)$ is a $l \times n$ matrix describing the muscular moment arms of the l muscles in u, derived from $\mathbf{U}$ (given pose q_t). In human characters, the number of muscles is much larger than the number of degrees of freedom ($l >> n$), making the number of solutions for u infinite. However, since our interest is to detect unrealistic muscle usage, we will seek a solution that assumes optimal load sharing between muscles. A common approach to achieve this is to minimize the sum of squared muscle stresses [vdH94]:

$$\underset{u}{\operatorname{argmin}} \sum_{i=1}^{l} \left(\frac{u_i}{u_{max,i}} \right)^2 \tag{4.13}$$

This optimization is subject to both the moment arm constraints in (4.12) as well as to $u_i \geq 0$ (muscles cannot produce a negative force). The values for u_{max} are defined in $\mathbf{U}$ and represent the maximum strength of the animated character's individual muscles. Common values of u_{max} are well-document in biomechanics literature and are mostly derived from cadaveric data [vdH94]. If desired, they can be adjusted to match the physique of any virtual character.

[1] When there are no ground reaction forces, equation (4.10) is equal to (4.4) and needs not be solved again.

Excess Muscle Force After we have determined muscle forces u for frame A_t, we wish to evaluate if they are realistic. Our quality measure λ is a dimensionless quantity, defined as the amount of excess muscle force, u_{ex}, normalized by the sum of the elements in u_{max}:

$$\lambda(A_t) = \frac{\sum_{i=1}^{l} u_{ex,i}}{\sum_{i=1}^{l} u_{max,i}} \quad , \quad u_{ex,i} = \begin{cases} u_i - u_{max,i} & \text{if } u_i > u_{max,i} \\ 0 & \text{if } u_i \leq u_{max,i} \end{cases} \tag{4.14}$$

where $u_{ex,i}$ is the excess muscle force produced by muscle i, based on the maximum force defined by $u_{max,i}$.

Final Quality Measures

We have shown how to compute quality measures $\kappa(A_t)$ and $\lambda(A_t)$ for any animation frame A_t. There are numerous ways to combine these individual scores to get a score for a full animation clip **A**. For our research, we have chosen to compute the average of all frames in the animation, which is sufficient when evaluating short animation clips.

4.3 Experiments

4.3.1 Musculoskeletal Experiments

To demonstrate our method, we investigate the effect of changing character body mass and animation speed on the physical realism of a set of motion captured animations. Changing the body mass of the character reflects a situation where a motion capture performer is much lighter or heavier than the virtual character. Some motions may not be physically realistic in that situation; we expect to see this reflected in muscle error λ, but not in dynamics error κ. Changing the weight of a character should not affect the degree in which its motion is subject to the Newton-Euler laws of motion. We expect that changing animation speed will affect both muscle error λ as well as dynamics error κ.

Our research will be based on the *Human Body Model*, which is a commercially available musculoskeletal model of the entire body [vdBGEZ*13]. The skeleton consists of 18 segments and 46 kinematic degrees of freedom. The muscle model includes 290 muscles. The inverse dynamics is solved using an approach described in detail by Kane et al. [KL96]. The squared muscle stress is minimized using the approach described by Xia and Feng [XF05], which is based on a recurrent neural network and has been developed with real-time performance in mind.

Animations are recorded using a 12 camera Vicon system and a custom 47 marker setup. In order to get smooth first and second order derivatives

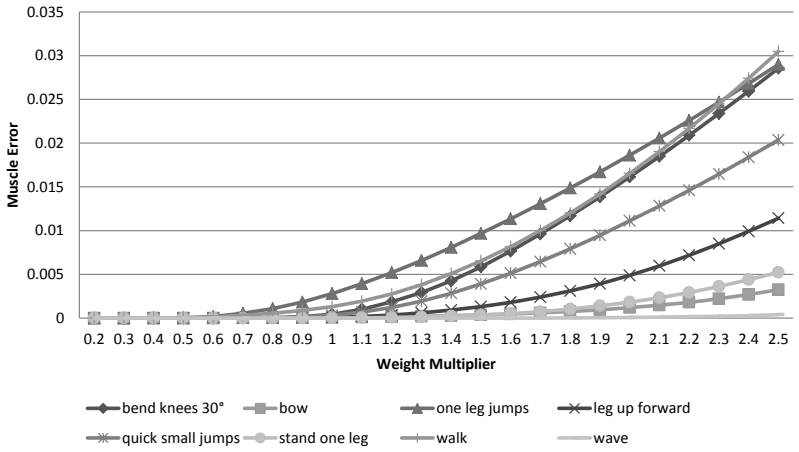


Figure 4.2: Muscle error λ, with varying subject weight

for q_t, the animation data is filtered using a second order Butterworth filter with a cut-off frequency of 4Hz. Our character ($m = 72$kg, $h = 1.70$m) has performed 15 different motions (see Figure 4.5). We separately vary the weight and speed multiplier in the range of 0.2 to 3.0 with steps of 0.2. We use a friction coefficient of $\mu = 1$.

4.3.2 Results

The results indicate that our measures behave as anticipated. First, there is a clear relation between body weight and muscle error (see Figure 4.2). This relation is stronger with more labor-intensive motions such as jumping or knee bending, and smaller for the more relaxed motions, such as waving.

We observed no significant change in the dynamics error as a result of weight change, which is in line with our expectations. The relation between speed change and muscle error is shown in Figure 4.3. The relation is stronger with highly dynamic motions, such as walking. With such motions, even slowing down the animation results in a slight increase in muscle error.

The effect of speed change on the dynamics error κ is shown in Figure 4.4. It can be seen that both slowing down and speeding up the animation leads to an increase in dynamics error, but only for dynamic motions such as walking and jumping. This shows it is not dynamically realistic to walk or jump at reduced or increased speed, without adjusting the motion trajectories.

Another interesting observation is that when evaluating a sped up walking animation, there is a relatively high increase in muscle error, compared

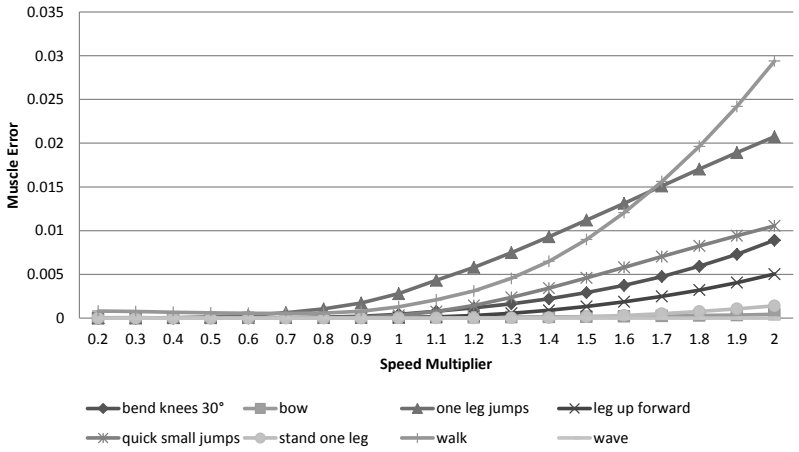


Figure 4.3: Muscle error λ, with varying animation speed

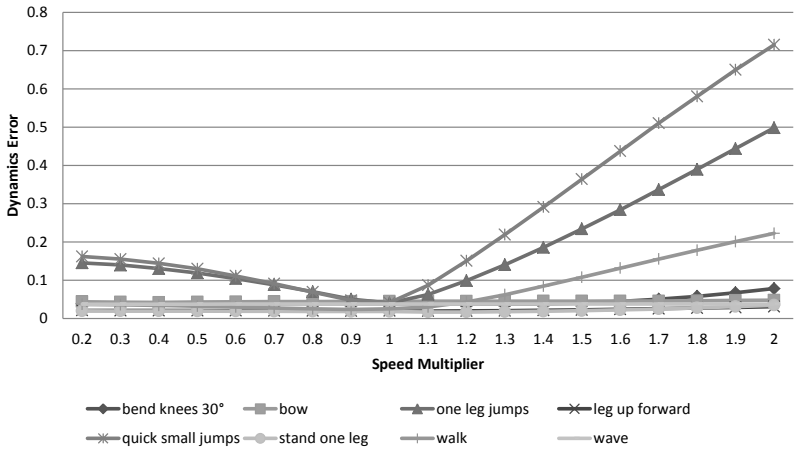


Figure 4.4: Dynamics error κ, with varying animation speed

to the increase in dynamics error. However, when evaluating a sped up jumping motion, there is a high increase in dynamics error, and a relatively low increase in muscle error. This could be interpreted as follows: it is physically more realistic for a character with supernatural strength to walk with increased speed than to jump with increased speed.

Finally, we show the maximum factor by which speed and weight can be

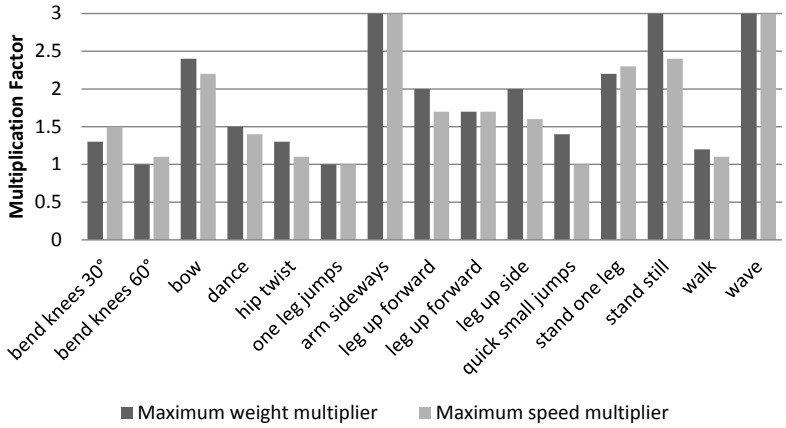


Figure 4.5: Maximum weight and speed multipliers for which $\lambda \leq 0.003$. Multipliers are capped at 3; this was the highest multiplier we tested for in our experiments.

increased before the muscle force required to perform a specific animation becomes physically unrealistic, based on an empirically-determined noise threshold (we use $\lambda > 0.003$). Figure 4.5 shows these limits for a number of animations.

4.4 Discussion

We have demonstrated a method for evaluating the physical realism of character animations using musculoskeletal models. We have shown how, after inverse dynamics calculation, ground reaction forces can be extracted from residual forces and moments. We have also shown how these residuals can be used to uniformly measure the dynamical validity of a motion, according to the Newton-Euler laws of motion. Finally, we have shown how the muscle forces estimated by a musculoskeletal model can be interpreted as a measure for realism of motion.

The results of our initial experiments are promising, and we see several applications for our quality measures. For example, the measures can be used to detect good transition points for motion graphs. Also, both measures can be shown as feedback to an animator during interactive motion editing. Excessive muscle force can even be visualized intuitively on a 3D mesh. For example, muscles can be shown in a specific color at moments where the required force is above a certain threshold.

Preliminary tests indicate that our estimated ground reaction forces

correspond well to measured data. A further investigation of this claim would require the recording of additional motions with synchronized force plate data. We are aware that our method is of limited use in biomechanics research, since it cannot detect situations where both feet exert horizontal forces in different directions. However, for the purpose of measuring physical realism this is not an issue.

Our method of estimating ground reaction forces can still be improved by incorporating a more advanced friction model. At this point, we assume all ground contact is static. With a dynamic friction model, it will be possible to detect foot sliding issues based purely on dynamics error and muscle error.

For some motions, we have found the muscle error to be above zero, even when neither weight nor speed was adjusted. We expect the reason for this is that some of the maximum muscle forces defined in our model are rather low in comparison to other models. We expect that adjusting these maximum muscle forces to more common values will resolve this issue.

Finally, an important next step in our research would be to evaluate the relationship between our quality measures and the perceived realism via user studies. It would then also make sense to compare our measures to those based on less detailed models, and analyze to what degree the enhanced accuracy of musculoskeletal models leads to a more accurate measure of perceived realism.

5

Injury Assessment for Physics-Based Characters

Severe physical trauma is a common ordeal for many virtual game characters. In fact, many games are designed around the concept of inflicting as much injury as possible on other characters. When such games aim for high realism, it is important that the assessment of injury is accurate.

In games that use kinematic animation systems, the possibilities for assessment are limited, because there exists no relation between the animation and real-world human injury data. Example indicators could be the initial height from which a character falls to the ground, or the impact velocity, but those ignore the specific forces acting on the character's individual body parts during impact. In animation systems based on simulated physics, however, all forces and torques that act on individual joints and body parts are readily available, and can be used to estimate injury in a way that is in line with physical reality.

There exist some games that model physical injury of physics-based characters. An example is *Stair Dismount* [`jet.ro/dismount`], a game in which the goal is to inflict as much damage as possible on a character by pushing it from a staircase. However, there exist no publications that explain what parameters and thresholds are used as a basis for these models, nor are there any published studies that show if and how such models correlate with perceived injury levels.

We propose an injury assessment model for physics-based virtual characters that is based on a comprehensive set of publications on human injury response levels. Our model produces a set of normalized measures that represent injury levels for several individual body parts. No tuning is required, as all parameters are directly derived from publications on injury research.

The fact that the individual measures are normalized allows straightforward aggregation into a single measure representing total injury. Our research includes a user study in which the output of our injury assessment model is compared to injury levels as perceived by human observers.

We believe there are several uses for our model. First, it can be used to measure injury of in-game physics-based characters; either to keep track of the overall health of the character, or to monitor the condition of specific body parts. Furthermore, we feel our measure can also be used for the development of motion controllers for physics-based characters. More specifically, our measure can be used as an optimization criterion for controllers that need to minimize physical injury, for instance while falling or blocking obstacles.

Our physics-based character model consists of rigid bodies and links, and our injury model is based on the forces, accelerations and velocities that these bodies and links are exposed to during simulation. We do not model effects such as tissue damage or local impact. As a result, our model is not suitable for measuring injuries such as gunshot wounds or cuts. The use of our model is limited to injuries from collisions with large, blunt objects that affect a relatively large portion of the body. Examples of such injuries are a character falling to the ground or from a staircase, a character in a vehicle hitting a wall.

This chapter is organized as follows. After first describing related work on injury measurement and animation perception, we describe in detail our injury assessment model, including sources for various model parameters. Next, we present methodology and results of the user study conducted to investigate the correlation with perceived injury levels. We conclude with a discussion and pointers for future research.

5.1 Related Work

Injury Assessment There exists a long history of research on human injury assessment, most of which is based on statistical data from real-life injuries [Pet81]. For the development of our measure we focus on research on injuries associated with motor-vehicle accidents. The *Abbreviated Injury Scale* (AIS) is a measure developed by the Association for Advancement of Automotive Medicine as a system for rating injuries in motor-vehicle accidents [GW06]. It consists of a dictionary of injury descriptions (mostly related to tissue or bone damage) and associated severity scores, and has been updated several times since its introduction. In 1984, General Motors published a set of reference values intended as guidelines for injury potential [Mer84], based on experiments with the *Hybrid III crash test dummy* [FKW77], which has been developed to represent the 50^{th} percentile male adult. Each of

these *Injury Assessment Reference Values* (IARVs) refers "to a human response level below which a specific significant injury is considered unlikely to occur for a given individual," which corresponds roughly with AIS ≥ 3. Most of these values consist of physical measurements and have a direct equivalent in physics simulation. Several later publications contain similar values for various body parts [MPI97, MME86, JWdC89, KBF*03, CKK*98, Zei84, MWB03]. Both the AIS and IARV focus on injuries to individual body parts. An adapted scale, called the *(New) Injury Severity Score* (ISS or NISS) defines how to combine IAS values from multiple traumas [BOHL74, SSGL*01].

Animation Perception Since the perception of injury is an important aspect of our research, it is also relevant to mention research on animation perception. Most well-known is the work of the group of O'Sullivan, including recent publications on plausibility of animated group conversations [EMO10], and on perception of animations subject to time-warping [PMO10]. Studies on physical reality of character animation include the work of Reitsma and Pollard [Rei03], who observe through user studies that errors in horizontal velocity and added accelerations in human jumping motions are easier observed by humans than errors in vertical velocity and added decelerations. In the previous chapter we have demonstrated a method to evaluate the physical realism of animations using a musculoskeletal model developed for research in biomechanics.

5.2 Assessment Method

In this section we describe how our injury measures are computed, using values from physical simulation. We will describe the general form of each measure, how each individual measure is computed, and how the individual measures can be combined into a single measure representing total injury.

5.2.1 Overview

Our injury assessment model consists of a set of individual measures, all of which represent an IARV-normalized maximum of a simulated variable averaged over a specific time window. The size of this time window is important, since the maximum of simulated variables such as acceleration can become very high in rigid body simulation. Luckily, most referenced research papers mention appropriate time windows. An overview of all measures is shown in Table 5.1.

Since all measures have a similar form (with the exception of the measure for head injury), we define the following function to simplify subsequent

Region	Physical Property	Unit	w	c	Measure(s)
Head	Linear Acceleration [MPI97]	HIC	0.015	700	D_H
Neck	Axial Compression [Mer84]	kN	0.030	4.0	D_{NC}
Neck	Axial Tension [Mer84]	kN	0.035	3.3	D_{NT}
Chest	Thoracic spine acceleration [MPI97]	G	0.003	60	D_C
Pelvis	Pelvis Acceleration [MME86, JWdC89]	G	0.003	130	D_{PE}
Arms	Compressive Wrist Loading [MWB03]	kN	0.010*	8.0	D_{W_L}, D_{W_R}
Legs	Femur Axial Force [Mer84]	kN	0.010	9.1	D_{FA_L}, D_{FA_R}
Legs	Femur Planar Force [KBF*03]	kN	0.003	3.9	D_{FP_L}, D_{FP_R}
Legs	Tibia Axial Force [CKK*98]	kN	0.010*	8.0	D_{T_L}, D_{T_R}
Legs	Compressive Ankle Loading [Mer84]	kN	0.010*	8.0	D_{A_L}, D_{A_R}
Feet	Foot Acceleration [Zei84]	G	0.010*	150	D_{P_L}, D_{P_R}

Table 5.1: Overview of injury measures for individual body regions. *Marked window sizes indicate estimates.

definitions:

$$D(f,w,c) = \max_{t_0} \left\{ \int_{t_0}^{t_0+w} \frac{1}{cw} f(t)dt \right\} \tag{5.1}$$

where $f(t)$ is a function describing an injury-specific physical property measured at time t, w is the window length, c is the IARV value used for normalization, and t_0 is a starting time of a time window instance. $D(f,w,c) \rightarrow [0,\infty\rangle$ represents the normalized maximum of the w-sized window-average of physical property $f(t)$, with a value of 1.0 representing the threshold of significant injury. Table 5.1 contains an overview of values used for w and c.

5.2.2 Individual Injury Measures

Head A common measure for head injury is the *Head Injury Criterion* (HIC), which is based on linear acceleration of the head [MPI97]. This measure differs slightly in form from our previously defined $D(f,w,c)$, since it is based on a non-linear relationship [HKL98]:

$$HIC(w) = \max_{t_0} \left\{ w \left[\frac{1}{w} \int_{t_0}^{t_0+w} \|a_H(t)\| dt \right]^{2.5} \right\} \tag{5.2}$$

in which w is the window time and $a_H(t)$ is the acceleration of the head at time t. A common window time is 15ms (also referred to as the HIC-15 score), and corresponding IARV is 700 [MPI97], which results in head injury measure D_H:

$$D_H = \frac{1}{700} HIC(0.015) \tag{5.3}$$

Neck Our neck injury measure is based on studies evaluating neck compression and tension [Mer84]. To measure these properties, we regard the axial constraint forces that are applied by the neck joint to the head and chest joint. If $\hat{F}_H(t)$ is the constraint force applied to the head, and $\hat{F}_C(t)$ the constraint force applied to the chest, then the neck injury measures D_{NC} and D_{NT} can be acquired using the dot product of each force and the Y-axis of the neck joint, y_N (see Figure 5.2), all in the global coordinate frame:

$$D_{NC} = D(\max(\hat{F}_H(t) \cdot y_N, 0) - \min(\hat{F}_C(t) \cdot y_N, 0), 0.030, 4.0) \quad (5.4)$$

$$D_{NT} = D(\max(\hat{F}_C(t) \cdot y_N, 0) - \min(\hat{F}_H(t) \cdot y_N, 0), 0.035, 3.3) \quad (5.5)$$

Chest Our chest injury measure is based on the maximum acceleration of the thoracic spine [MPI97]. We acquire chest injury measure D_C by using the acceleration of the chest segment of our virtual character, $a_C(t)$:

$$D_C = D(\|a_C(t)\|, 0.003, 60) \quad (5.6)$$

Pelvis Our pelvis injury measure is based on the maximum acceleration of the pelvis during side impact, based on biomechanics research by Morgan et al. [MME86] and Janssen et al. [JWdC89]. We acquire our pelvis injury measure D_{PE} using the acceleration of the virtual pelvis segment, $a_{PE}(t)$:

$$D_{PE} = D(\|a_{PE}(t)\|, 0.003, 130) \quad (5.7)$$

Arms Muller et al. [MWB03] investigate the maximum force that can be applied to the wrist before fracture. To acquire wrist injury D_W, we use the total magnitude of the constraint forces applied by the wrist joint to hand $\hat{F}_{HA}(t)$ and lower arm $\hat{F}_{LA}(t)$. Muller et al. mention no time window; based on averages from Mertz et al. [Mer84] we decided upon using a 10ms window. The measure then becomes (computed separately for left wrist, D_{W_L}, and right wrist, D_{W_R}):

$$D_W = D(\|\hat{F}_{HA}(t)\| + \|\hat{F}_{LA}(t)\|, 0.010, 8.0) \quad (5.8)$$

Legs There exist several different publications on IARV scores for legs. These include values for femur axial force (F_{FA}) [Mer84], femur planar force (F_{FP}) [KBF*03] and tibia axial force (F_{TA}) [CKK*98]. We acquire these forces by taking the magnitude of the acceleration of the corresponding virtual body segment in a specific direction in the local coordinate frame, multiplied by the body segment mass. The resulting injury measures, D_{FA}, D_{FP} and D_T,

for each individual leg, then become:

$$D_{FA} = D(\|F_{FA}(t)\|, 0.010, 9.1) \tag{5.9}$$
$$D_{FP} = D(\|F_{FP}(t)\|, 0.003, 3.9) \tag{5.10}$$
$$D_{T} = D(\|F_{TA}(t)\|, 0.010, 8.0) \tag{5.11}$$

Ankles Mertz [Mer84] has published data on maximum ankle load. We compute the corresponding ankle damage score, D_A, similar to wrist damage, using constraint forces on lower leg, $\hat{F}_{TA}(t)$, and foot, $\hat{F}_P(t)$ (individually for left and right):

$$D_A = D(\|\hat{F}_{TA}(t)\| + \|\hat{F}_P(t)\|, 0.010, 8.0) \tag{5.12}$$

Feet Zeidler [Zei84] has suggested the IARV score for foot acceleration, $a_P(t)$, to be $150G$. We again use an estimated window size of 10ms to compute our foot injury measure D_P (individually for left and right foot):

$$D_P = D(\|a_P(t)\|, 0.010, 150) \tag{5.13}$$

5.2.3 Combining Individual Measures

We have so far defined a set $\mathbf{M}$ of individual injury measures, each describing the amount of injury of a specific body part (see Table 5.1). Since each value is normalized to represent overall severity (with a value of 1 representing the threshold of serious injury), we can meaningfully combine the individual measures by using the average, D_{AVG}:

$$D_{AVG} = \sum_{D \in \mathbf{M}} \frac{1}{\|\mathbf{M}\|} D \tag{5.14}$$

Alternatively, one can choose to only use the three highest scores for this average, inspired by the *New Injury Severity Score* (NISS) [SSGL*01], or to use a quadratic relation, as suggested by [BOHL74]. However, our results do not support the idea that this will lead to better estimates of perceived total injury.

5.3 Experiments

To evaluate our injury assessment model, we have conducted a user study to investigate if the output of our model correlates with injury levels of virtual characters as perceived by human observers. This section describes the experiment setup and results.

5.3.1 Trial Data

Our trial data consisted of 30 short (4 second) video clips of a virtual character falling in a physically simulated environment. In each trial, the virtual character falls down from a random initial orientation and height. The initial orientation was generated by applying random rotations around the X and Z axis, while the initial height of the center of mass (COM) was varied between $1.5m$ and $4.5m$. An example fragment of a trial video clip is shown in Figure 5.1.

Figure 5.1: Example fragment of animation sequence shown as part of the survey.

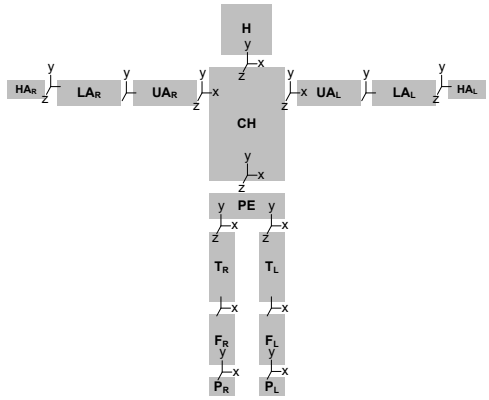
We used the *Open Dynamics Engine* (ODE) [Smi06] to perform the physics simulation that generated the animation, using an integration time step of 0.0003s. We utilized ODE's *Error Reduction Parameter* (ERP) and *Constraint Force Mixing* parameter (CFM) to emulate the effect of shock absorption due to soft tissues, joint compliance and ligaments (ERP $= 0.5$, CFM $= 0.001$; see [Smi06] for details). The hierarchy and degrees-of-freedom (DOFs) of our virtual character are shown in Figure 5.2. Low-gain PD-controllers were used to drive the character joints towards their initial T-stance position ($k_p = 10$, $k_d = 2$), with torque maximum set to 1000Nm for each joint.

5.3.2 User Study

The trial data was presented to 34 individuals (21 male, 13 female), with ages between 23 and 35 years. Each individual was shown all clips in random order; they were instructed to assign each clip a value between 1 (minimum) and 7 (maximum), based on how much total physical damage they felt was inflicted upon the entire body of the virtual character. Furthermore, they were informed that a score of 4 represents the threshold of significant injury. This resulted in a total of $N = 34 \times 30 = 1020$ evaluations.

5.3.3 Results

We have found a significant correlation between injury measure D_{AVG} and the corresponding user score S: $\mathrm{corr}(D_{AVG}, S) = 0.603$, with $N = 1020$ and $p < 0.001$. Figure 5.3 displays a scatter plot of the computed D_{AVG} for each



Body	Body part	Shape	Position (X, Z, Y)	Dimensions
PE	Pelvis	box	(1.20, 0.00, 1.00)	X=0.30, Z=0.25, Y=0.20
CH	Torso	box	(1.20, 0.00, 1.30)	X=0.25, Z=0.20, Y=0.50
H	Head	sphere	(1.20, 0.00, 1.75)	R=0.10
T_R	Upper leg	capsule	(1.29, 0.00, 0.68)	R=0.06, L=0.35
T_L	Upper leg	capsule	(1.11, 0.00, 0.68)	R=0.06, L=0.35
F_R	Lower leg	capsule	(1.29, 0.00, 0.25)	R=0.06, L=0.30
F_L	Lower leg	capsule	(1.11, 0.00, 0.25)	R=0.06, L=0.30
P_L	Foot	box	(1.11, -0.07, 0.03)	X=0.10, Z=0.35, Y=0.05
P_R	Foot	box	(1.29, -0.07, 0.03)	X=0.10, Z=0.35, Y=0.05
UA_L	Upper arm	capsule	(0.88, 0.00, 1.50)	R=0.05, L=0.35
UA_R	Upper arm	capsule	(1.52, 0.00, 1.50)	R=0.05, L=0.35
LA_L	Lower arm	capsule	(0.50, 0.00, 1.50)	R=0.05, L=0.30
LA_R	Lower arm	capsule	(1.90, 0.00, 1.50)	R=0.05, L=0.30
HA_L	Hand	box	(0.25, 0.00, 1.50)	X=0.15, Z=0.08, Y=0.03
HA_R	Hand	box	(2.15, 0.00, 1.50)	X=0.15, Z=0.08, Y=0.03

Figure 5.2: Character DOFs, body segments, and corresponding dimensions.

trial, and the corresponding averages of user score S. In comparison, the correlation between initial COM height H and user score S is $\text{corr}(H,S) = 0.466$, with $N = 1020$ and $p < 0.001$. This supports the hypothesis that our measure is a more accurate representation of perceived injury than falling height, when regarding characters falling. An overview of the average of D_{AVG} for each individual user score is shown in Figure 5.4, affirming the correlation between the two.

If we only regarded the top 3 most occurring injuries (similar to the NISS scoring system), the correlation decreases to 0.522 ($p < 0.001$). The use of quadratic terms (as in the NISS scoring) only further decreases the correlation. This means our results do not support the use of NISS-related methods for computing overall injury.

Figure 5.5 shows the average value for each of the individual measures. It can be seen that neck compression injury D_{NC} is the measure with the highest average score. This is mostly due to the fact that some trials consisting of

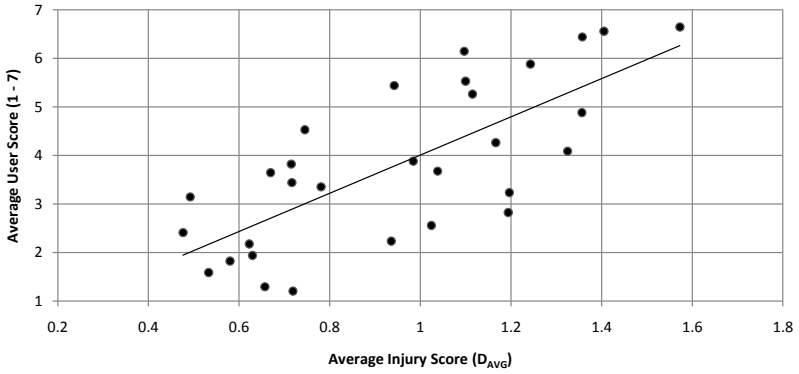


Figure 5.3: Scatter plot of injury measure D_{AVG} for each trial (horizontal) and the average of user score S (vertical). $\text{corr}(D_{AVG}, S) = 0.603$, $N = 1020$, $p < 0.001$

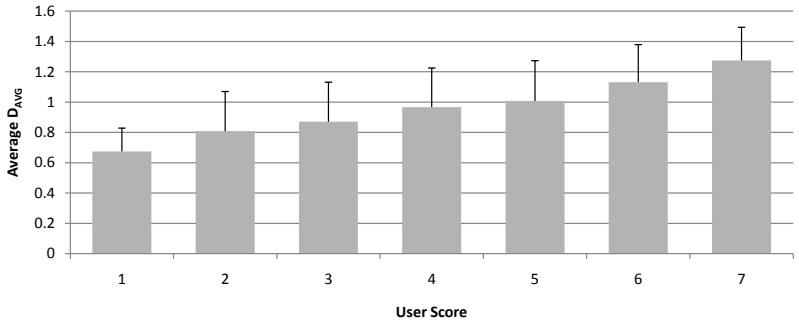


Figure 5.4: Average of injury measure D_{AVG} for each user score.

head-first landing animations had extremely high neck compression. This also explains the high standard deviation of D_{NC} in Figure 5.5.

Results on correlation studies for individual injury measures show that most measures have a significant positive correlation with perceived injury, while some demonstrate a stronger correlation than others. Measures for ankle compression (D_{A_L} and D_{A_R}) show a negative correlation. This can be explained by the fact that ankle compression is highest when a character lands on his feet, while observers generally assigned relatively low injury scores to such animations.

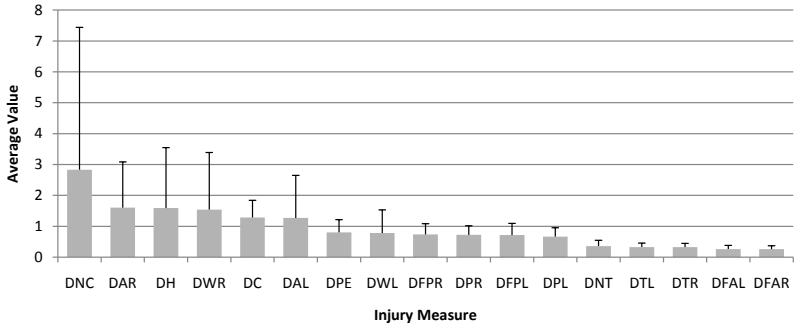


Figure 5.5: Average value for each individual injury measure. A value of 1 represents the threshold of serious injury for a specific body part.

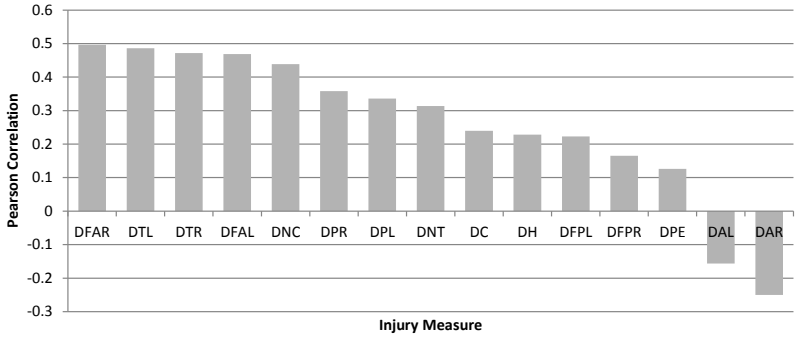


Figure 5.6: An overview of the correlations between individual injury measures and perceived injury by human observers. All correlations are significant ($p < 0.001$) except the correlations for wrist injury (D_{W_L} and D_{W_R}), which are excluded from the figure.

5.4 Discussion

We have presented a model for computing injury measures for physics-based characters, based on research on human injury response levels. We have found a significant correlation between the average injury measure and the total injury level as perceived by human observers. This correlation is stronger than the correlation between perceived injury and fall height (0.603 versus 0.466, respectively, with $N = 1020$ and $p < 0.001$).

There are several limitations to our injury assessment model. It does not include key body parts such as knees, elbows and shoulders, because we

have not been able to find publications that contain suitable data for these areas. However, we do believe that these omissions have not disrupted our research too much, since relevant trials had significant injury in neighboring body parts. If desired, proper estimates may be derived from other measures, without the need for published data. In addition, it may be possible to assess specific facial injuries by using location-aware impact measures. In order to accurately measure injury types such as shot wounds or cuts, we suggest using a more advanced model that includes tissue damage and topology of vital organs.

Perception studies often leave room for debate, and ours is no different. First, we used a very basic looking character model; a more detailed model may have triggered different responses. Second, we have only tested for straight falling motions; other motions (falling from stairs, collisions with large objects, etc.) may also have triggered different responses. Several of our test subjects reported that some animations appeared unrealistic, because characters bounced where they normally would break (we did not allow any joint dislocation in our simulation). There are many additional appearance factors that complicate our perception study, such as viewing angle, playback speed, character model, background color, etc. However, test results indicate our test material was at least partly representative.

The work presented in this chapter is an initial attempt at injury assessment for physics-based virtual characters. There is much room for improvement, while additional user studies can create better insights on how injury of animated characters is perceived. As mentioned in our introduction, injury is an important aspect of gaming, which justifies further research in this area.

Part III

Physics-Based Character Animation

6

SIMPLE DATA-DRIVEN CONTROL FOR SIMULATED BIPEDS

Despite widespread adoption of physics simulation for lifeless phenomena, such as cloth, water or *rag-doll* characters, production games still resort to kinematics-based approaches for the animation of actively controlled characters.

In an attempt to understand this reservation, we distinguish between two different approaches used in physics-based character animation research. On one hand, there are methods that compute joint torques based on *kinematic parameters*, such as joint angles or center-of-mass. These methods are relatively easy to implement and integrate well with common physics engines, such as the *Open Dynamics Engine* or *PhysX*. Recent publications that use this approach have displayed a variety of robust and versatile locomotion controllers (*e.g.* [CBvdP10, WFH10]). However, these methods allow style control only through *key-framing* or optimization for high-level criteria. Methods that use captured reference motions have several limitations or

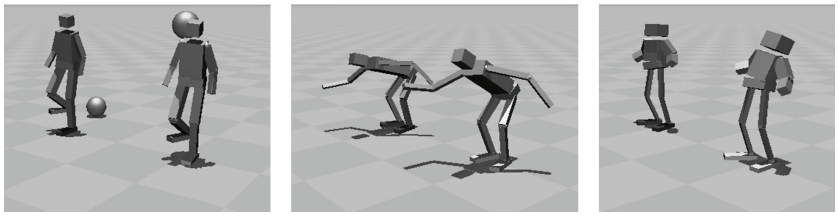


Figure 6.1: Example results of the control framework.

require extensive preprocessing (*e.g.* [SKL07, YLvdP07]).

On the other hand, there are methods that compute joint torques based on the *equations-of-motion* that describe the underlying character dynamics. Such methods have demonstrated impressive results, including the ability to robustly track recorded running and turning motions, as well as robust navigation over uneven terrain (e.g. [MLPP09, WP10, MdLH10]). However, these methods are also much more difficult to implement. They require thorough knowledge of constrained dynamics and optimization theory and do not mix trivially with common physics engines. In addition, computational requirements are often high: publications typically report real-time performance for only a single character at a time on a modern PC.

Based on these observations, we expect the former category to be more eligible for adoption in industry at this point in time – especially for the rapidly developing mobile application market. Our contribution consists of a control framework that can robustly track unmodified reference motions based on kinematic properties, without the need to explicitly invert the equations-of-motion. It advances the state-of-the-art by allowing robust control based on reference motions that contain a diversity and agility previously unseen in comparable methods.

Our control framework consists of three components: *Proportional-Derivative Control* to mimic motion characteristics, a specific form of *Jacobian Transpose Control* for balance control, and *Covariance Matrix Adaptation* for off-line parameter optimization based on a novel high-level reward function. It produces simulations at approximately 4× real-time on a single core of a modern PC. We demonstrate the capabilities of our framework through a variety of motions, including squatting, bowing, kicking and dancing, as well as display its ability to withstand external perturbations and adapt to changes in character morphology.

6.1 Related Work

Following the above introduction, we distinguish between publications that compute joint torques based on *kinematic parameters* and those that compute torques based on the *equations-of-motion* describing the underlying character dynamics. A detailed review of approaches belonging to the first category can be found in Sections 3.1 and 3.3, while approaches belonging to the latter category can be found in Section 3.2.

Torques Computed from Kinematic Properties Early examples using this approach include the work of Raibert and Hodgins [RH91], as well as the milestone human athletics controller by Hodgins et al. [HWBO95]. Both techniques control speed and balance through on-line modulation

of target trajectories, and compute torques using Proportional-Derivative Control. Other key examples are the walking controller of Laszlo et al. [LvdPF96], the various controllers of Faloutsos et al. [FvdPT01], and the interactive tracking controller of Zordan and Hodgins [ZH02]. More recently, Yin et al. [YLvdP07] introduced SIMBICON, a framework supporting various types of robust gaits, also using on-line trajectory modulation for balance and PD Control, which has been the basis for many subsequent publications [CBYvdP08, CBvdP09, YCBvdP08, WFH09, WFH10]. Coros et al. incorporate a form of *Jacobian Transpose Control* [SADM94], resulting in versatile and generic locomotion controllers for bipeds [CBvdP10] and quadrupeds [CKJ*11]. Their control method is based on the *Virtual Model Control* framework by Pratt et al. [PCTD01] (see also Section 6.2.2).

To minimize manual tuning, several control frameworks use off-line parameter optimization based on high-level criteria (a topic famously explored by Sims [Sim94]). Hodgins and Pollard [HP97] show how optimization can be used to adapt to changes in character morphology. The parameters of the SIMBICON framework have been optimized to generate new behaviors [YCBvdP08], to control style [WFH09], and to increase robustness [WFH10]. Tan et al. [TGTL11] use off-line parameter optimization for swimming controllers. Recent publications typically use Covariance Matrix Adaptation (CMA) [Han06] for off-line parameter optimization.

There are not many publications that use kinematics-based torque computation in combination with captured reference motions. An early exception is the work of Zordan and Hodgins [ZH02], who track several balanced dual-stance motions using a balance compensation strategy based on the work of Wooten and Hodgins [WH00]. Sharon and Van de Panne [SvdP05] and Sok et al. [SKL07] demonstrate data-driven controllers that are limited to 2D, both based on state-action maps. The SIMBICON framework can also be used in combination with captured reference motions, but this requires unautomated pre-processing of the motion data [YLvdP07]. Subsequent work of Lee et al. [LKL10] does demonstrate robust tracking of unaltered reference motions using a SIMBICON-like balance strategy, but their method uses inverse dynamics for torque computation, which requires access to the equations-of-motion.

Torques Computed from Equations-of-Motion This category of approaches is linked to the *spacetime* animation framework by Witkin and Kass [WK88] and computes torques through constrained optimization based on the equations-of-motions describing the character dynamics. Stewart and Cremer [SC92b] use this method to produce physically correct animations for climbing and descending stairs. Abe et al. [AdSP07] use quadratic programming (QP) to find the set of torques that optimally drive the center-of-mass over the base of

support, demonstrating robust balance behaviors on moving bases. Macchietto et al. [MZS09] use inverse dynamics to track trajectories that minimize angular momentum, to which Wu and Zordan [WZ10] add stepping motions for additional balance correction. Lee et al. [LKL10] use inverse dynamics in combination with a SIMBICON-like balance strategy to robustly track captured reference motions. De Lasa et al. [dLMH10] use on-line prioritized optimization to construct robust locomotion and jumping controllers. Muico et al. [MLPP09, MPP11] use off-line optimization of reference motions and contact forces, in combination with a nonlinear quadratic regulator to create agile walking and running controllers. Wu and Popović [WP10] use optimized end-effector trajectories in combination with QP for navigation over uneven terrain.

Other publications use short-horizon optimization of an internal model to acquire an on-line look-ahead policy. Examples are the work of Da Silva et al. [dSAP08b], who demonstrate walking controllers based on reference motions, and Kwon and Hodgins [KH10], who demonstrate running controllers based on reference motions. Mordatch et al. [MdLH10] demonstrate locomotion over constrained and uneven terrain, using the low-level control framework of [dLMH10]. Jain et al. [JL11b] perform motion tracking using a model based on the principal modes of the character, while Ye and Liu [YL10] use an abstract internal model based on center-of-mass and ground reaction force.

Because of the tight link between control and simulation in these methods, physics simulation is mostly performed as part of the control framework. Exceptions of publications in which such methods have been integrated with common physics engines are the work of Da Silva et al. [dSAP08b] (*Open Dynamics Engine*) and the uneven terrain controller of Wu and Popović [WP10] (*PhysX*).

Our Work Our control framework uses a combination of Proportional-Differential Control for tracking, Jacobian Transpose Control for balance, and Covariance Matrix Adaptation for off-line parameter optimization. Even though each of these techniques has been used in several related publications, they have not been used in this specific combination, and not for the purpose of tracking unaltered reference motions.

In spirit, our work is most closely related to the motion capture driven controllers of Zordan and Hodgins [ZH02], the locomotion controllers of Coros et al. [CBvdP10], which make extensive use of virtual forces, and to the work of Lee et al. [LKL10], who perform impressive tracking of a wide range of unaltered captured walking motions, using a balance correction algorithm based on kinematic properties. The main difference between our work and that of Zordan and Hodgins [ZH02] is that their controllers are limited to dual

stance motions with upper-body dynamics, while we demonstrate motions of both single and double stance, with significant lower-body dynamics. The main difference between our work and that of Coros et al. [CBvdP10] is that in their framework, motion trajectories are constructed through key poses, while our framework uses unaltered reference motion trajectories. An important difference between our work and that of Lee et al. [LKL10] is that their method requires inverse dynamics for torque computation. Inverse dynamics is not supported by most common physics engines, and the required inversion of the equations-of-motion has additional computational requirements. In addition, inverse dynamics control inefficiently fights the natural dynamics of a physics-based character [PCTD01], which may lead to unnatural joint compliance. Another key difference between our work and that of both Coros et al. [CBvdP10] and Lee et al. [LKL10] is that their balance strategies are largely based on swing foot placement for balance – inspired by the SIMBICON balance strategy of Yin et al. [YLvdP07]. This makes their control frameworks suitable for locomotion, while the static balance algorithms used in our framework make it more suitable for in-place motions.

6.2 Control Framework

The goal of our control framework is to have a simulated biped follow an unmodified reference motion as faithfully as possible, while maintaining balance and withstand external perturbations. It consists of the following elements:

- *Tracking control* (Section 6.2.1). To mimic the characteristics of a recorded motion, we track the reference trajectories of the individual degrees of freedom (DOFs) using Proportional-Derivative (PD) Control.

- *Balance control* (Section 6.2.2). To maintain balance, we use a specific form of Jacobian Transpose (JT) Control, consisting of a set of *virtual forces* and *virtual torques*.

- *Off-line parameter optimization* (Section 6.2.3). To find the right set of control parameters for a specific motion or character, we perform off-line parameter optimization based on high-level objectives, using Covariance Matrix Adaptation.

At any time t, we assume availability of the values of all degrees of freedom and their first order derivatives, both for a simulated character, C, and a reference motion, A. For a simulated character, DOFs and their

derivatives are accessible from the physics simulation engine. For a reference motion, the first order derivatives can be acquired through low-pass filtering and differentiation.

6.2.1 Tracking Control

To mimic the local characteristics of a captured motion, our control framework uses Proportional-Derivative (PD) Control for tracking reference trajectories of the individual DOFs. PD Control attempts to minimize the displacement between a reference joint angle, θ_A, and the corresponding joint angle of the simulated character, θ_C, as well as the difference between reference joint velocity, $\dot{\theta}_A$, and simulated joint velocity, $\dot{\theta}_C$. The torque produced by PD Control, τ_{pd}, is linearly proportional to both these differences:

$$\tau_{\mathrm{pd}} = \mathrm{k_p}(\theta_A - \theta_C) + \mathrm{k_d}(\dot{\theta}_A - \dot{\theta}_C) \tag{6.1}$$

The responsiveness to deviations in position and velocity is controlled through gains $\mathrm{k_p}$ and $\mathrm{k_d}$. See also Section 3.1.1 for more details on PD control and controller gains. In this research, $\mathrm{k_p}$ and $\mathrm{k_d}$ are determined individually for each actuated DOF through off-line optimization (see Section 6.2.3).

6.2.2 Balance Control

Since PD control tracks only actuated degrees of freedom, additional control is required to prevent errors in global position and orientation to accumulate, resulting in loss of balance (see also [SKL07]). Our balance control strategy is based on the application of *virtual forces* and *virtual torques*, both of which are forms of *Jacobian Transpose Control*.

Jacobian Transpose Control

The essence of this control method is that it enables control in Cartesian space for linked structures. Using the Jacobian Transpose, it is possible to compute the set of torques that emulate the effect of an external force or torque, applied to a specific body or a virtual point (such as the center-of-mass). Virtual forces and torques are applied to a *chain of linked bodies*, starting from a static base link (such as the stance foot) and moving to a target link (see Figure 6.2 for an illustration). The set of joint torques τ_F that emulate a virtual force F applied at point p corresponds to:

$$\tau_F = J(p)^T F \tag{6.2}$$

where $J(p)$ is the Jacobian that represents the rate of change of a point p for each DOF i connecting the targeted chain of bodies. Similarly, it is

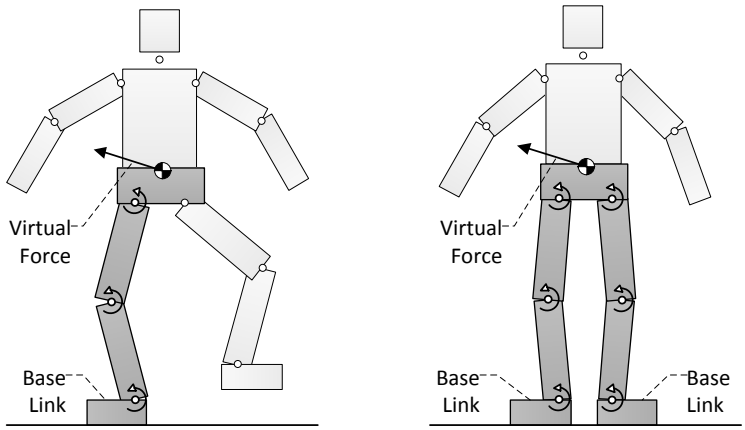


Figure 6.2: Virtual forces applied to the center of mass, through a chain of linked bodies, for single stance (left) and dual stance (right).

possible to apply a virtual torque T to a specific body at the end of a chain. If the orientation of the target link is represented using an *exponential map* [Gra98], $e \in \mathbb{R}^3$, then the set of joint torques that emulate a virtual torque T can be calculated using the Jacobian that represents the rate of change of e for each DOF i in the chain of bodies:

$$\tau_T = J(e)^T \; T \tag{6.3}$$

Details on how to acquire $J(p)$ and $J(e)$ are provided in Section 3.1.2. Note that, since bipeds contain no links that are truly static, the effect of a virtual force or torque is always an approximation and can be determined only after simulation. That said, we have found this approximation to be sufficiently accurate to work well in practice.

Stance State

The selection of the chain of bodies to which we apply our virtual forces and torques depends on the *stance state* of the character. A stance state S can be one of the following:

$$S \in \{\text{left\_stance}, \text{right\_stance}, \text{dual\_stance}, \text{flight}\} \tag{6.4}$$

The stance state must be computed separately for simulated character and reference motion. For the simulated character, the state can be derived from contact state in the physics engine, for the reference motion we base the state on the height of the ankle joint position.

Base of Support

In our framework, the base of support is represented by a point p_{base}, which is based on the stance state and the projected ankle joint position(s):

$$p_{\text{base}} = \begin{cases} \perp p_{\text{ankL}} & \text{if } S_A = \text{left\_stance} \\ \perp p_{\text{ankR}} & \text{if } S_A = \text{right\_stance} \\ \frac{\perp p_{\text{ankL}} + \perp p_{\text{ankR}}}{2} & \text{otherwise} \end{cases} \tag{6.5}$$

where $\perp p_{\text{ankL}}$ and $\perp p_{\text{ankR}}$ are the projected positions of left and right ankle joint, respectively, on the ground plane. To allow comparison between the base position of the reference motion, $p_{\text{base,A}}$, and the base position of the simulated character, $p_{\text{base,C}}$, both must be computed using the same stance state. We therefore always determine the base of support using the stance state of the reference motion, S_A, even if the character state, S_C, is different.

Balance Strategy Components

Our balance control strategy consists of a combination of virtual forces and torques, each targeting a different aspect of balance:

- A virtual force applied to the center-of-mass (COM) of the character, to minimize differences in COM position and velocity between the simulated character motion and the reference motion.
- A virtual torque applied to the pelvis, to maintain the upper-body posture of the simulated character.
- A virtual torque applied to the pelvis, to regulate the total angular momentum of the character.

Each of these components will be described in detail in the upcoming paragraphs.

Center-of-Mass Position and Velocity The first component of our balance control strategy compensates for differences in COM position and velocity between the simulated character motion and the reference motion. It does so by applying a virtual force at the COM position of the simulated character, to the chain of bodies from stance foot to pelvis (see Figure 6.2). For a character consisting of k bodies, COM position p_{com} and COM velocity v_{com} are a weighted average of individual body positions, p_i, and velocities v_i:

$$p_{\text{com}} = \sum_{i=1}^{k} \frac{\text{m}_i}{\text{m}_{\text{tot}}} p_i \quad , \quad v_{\text{com}} = \sum_{i=1}^{k} \frac{\text{m}_i}{\text{m}_{\text{tot}}} v_i \tag{6.6}$$

in which m_i is the mass of body i and m_{tot} is the total mass. Mass properties are derived from body geometry, both for the simulated character and the reference motion.

We attempt to control balance by regarding the COM position of a character with respect to its base of support. This relative position $\hat{p}_{com}$ corresponds to:

$$\hat{p}_{com} = p_{com} - p_{base} \tag{6.7}$$

The virtual force F_{com} that minimizes the difference between relative COM position of the reference motion, $\hat{p}_{com,A}$, and simulated character, $\hat{p}_{com,C}$, as well as the difference in COM velocity between reference motion, $v_{com,A}$, and simulated character, $v_{com,C}$, now becomes:

$$F_{com} = w_{cp}(\hat{p}_{com,A} - \hat{p}_{com,C}) + w_{cv}(v_{com,A} - v_{com,C}) \tag{6.8}$$

in which w_{cp} and w_{cv} are constants controlling force magnitude. These constants are determined through off-line optimization (see Section 6.2.3). Different parameter sets are used depending on the stance state of the character (single stance or dual stance). The corresponding set of individual joint torques, τ_{com}, which are applied to each DOF in *stance ankle*, *stance knee* and *stance hip* can be acquired using Equation (3.5).

Trunk Orientation To maintain the posture of the upper body, we attempt to minimize the difference in trunk orientation between the simulated character motion and the reference motion. We do so by applying a virtual torque to the chain of bodies from stance foot to the pelvis (see Figure 6.2). If q_A is a quaternion describing the trunk orientation of the reference motion, and q_C that of the simulated character, then the virtual torque T_{trunk} corresponds to:

$$T_{trunk,i} = w_{to}\, q_{trunk,C} \exp(q_{trunk,A}\, q^{-1}_{trunk,C}) \tag{6.9}$$

where $\exp : \mathbb{R}^4 \Rightarrow \mathbb{R}^3$ is a function that extracts an *exponential map* from a quaternion (see [Gra98]), and w_{to} is a constant controlling the magnitude of the torque, determined through off-line optimization (see Section 6.2.3). The corresponding set of individual joint torques, τ_{trunk}, which are applied to each DOF in *stance ankle*, *stance knee* and *stance hip* can be acquired through Equation (3.7).

Angular Momentum Regulation of the angular momentum (AM) is an important aspect of biped balance control (see also [MZS09]). For a character consisting of k bodies, the angular momentum, L, corresponds to:

$$L = \sum_{i=1}^{k} m_i(p_i - p_{com}) \times v_i + R_{world,i} I_i \omega_i \tag{6.10}$$

in which m_i is the mass of body i, p_i its position, v_i its linear velocity, and ω_i its angular velocity. I_i is the 3×3 *inertia tensor* matrix, describing the mass distribution of body i, while $R_{world,i}$ is a rotational matrix from local frame of body i to the world coordinate frame. Mass properties are derived from body geometry, both for the simulated character and the reference character.

To minimize difference in angular momentum between a reference motion, L_A, and simulation, L_C, we apply a virtual torque T_{am} to the chain of bodies from stance foot to pelvis:

$$T_{am} = w_{am}(L_A - L_C) \tag{6.11}$$

in which w_{am} is a constant controlling magnitude, determined through off-line optimization (see Section 6.2.3). The corresponding set of individual joint torques, τ_{am}, which are applied to each DOF in *stance ankle*, *stance knee* and *stance hip* can be acquired through Equation (3.7).

Combining the Individual Components

After all joint torques are computed using the tracking control and balance control algorithms, they can be added together to a single torque vector:

$$\tau_{control} = \tau_{pd} + \tau_{com} + \tau_{trunk} + \tau_{am} \tag{6.12}$$

Following the work of Wang et al. [WFH10], we add *motor noise* to our control output, in the form of random torque perturbations, τ_{noise}. The application of motor noise can provide robustness against unmodeled phenomena, including numerical errors during simulation. We use a simplified motor noise model, which consists of individual values ranging from -5 to 5 Nm, randomly sampled each frame from a uniform distribution. Unlike the model of Wang et al., our noise does not increase with higher torques. However, our noise range is in the same order of magnitude as theirs for average torque levels ($\tau \approx 20$Nm).

For each DOF i, we also enforce a maximum torque value $\tau_{max,i}$, resulting in the following final torque τ_i:

$$\tau_i = \begin{cases} \tau_{max,i} + \tau_{noise,i} & \text{if } \tau_{control,i} > \tau_{max,i} \\ -\tau_{max,i} + \tau_{noise,i} & \text{if } \tau_{control,i} < -\tau_{max,i} \\ \tau_{control,i} + \tau_{noise,i} & \text{otherwise} \end{cases} \tag{6.13}$$

6.2.3 Off-line Parameter Optimization

In this final step of our method, we optimize the various parameters of our control framework, based on high-level optimization criteria. The set of

parameters we wish to optimize consists of the *PD Controller gains* and the *weights from our balance control strategy*:

$$\mathrm{K} = \left\{ \mathrm{k}_{\mathrm{p},1}, \ldots, \mathrm{k}_{\mathrm{p},n} \ , \ \mathrm{k}_{\mathrm{d},1}, \ldots, \mathrm{k}_{\mathrm{d},n} \right\} \tag{6.14}$$

$$\mathrm{W}_{\mathrm{single}} = \left\{ \mathrm{w}_{\mathrm{cv}}, \mathrm{w}_{\mathrm{cp}}, \mathrm{w}_{\mathrm{to}}, \mathrm{w}_{\mathrm{am}} \right\} \tag{6.15}$$

$$\mathrm{W}_{\mathrm{dual}} = \left\{ \mathrm{w}'_{\mathrm{cv}}, \mathrm{w}'_{\mathrm{cp}}, \mathrm{w}'_{\mathrm{to}}, \mathrm{w}'_{\mathrm{am}} \right\} \tag{6.16}$$

For the parameters in K, we use n different values for $\mathrm{k_p}$ and $\mathrm{k_d}$, one for each DOF, disregarding symmetric DOFs (we use the same $\mathrm{k_p}$ and $\mathrm{k_d}$ for left and right sided versions). For the weights in our balance compensation strategies, we use different sets for single stance, $\mathrm{W_{single}}$, and dual stance, $\mathrm{W_{dual}}$, depending on the state of the simulated character, S_C.

For a character with n unmirrored DOFs, the total set of parameters we wish to optimize, P, becomes:

$$\mathrm{P} = \left\{ \mathrm{K}, \mathrm{W}_{\mathrm{single}}, \mathrm{W}_{\mathrm{dual}} \right\} \quad , \quad \|\mathrm{P}\| = 2n + 8 \tag{6.17}$$

Optimization Objective The goal of our optimization is to find the set of parameters for which the simulated character C tracks the reference motion A most faithfully. To determine this, we use the following error measures:

- *Pose displacement* (E_{pose}). The pose of the simulated character should not deviate too much from the reference motion. An evident case in which this occurs is when the simulated biped has lost its balance. We define pose displacement, $E_{\mathrm{pose}}(t) \rightarrow \mathbb{R}$, as the weighted average of the displacement of the individual bodies, with respect to the COM of the character.

- *Stance state error* (E_{stance}). Sometimes the stance state of the simulated character is different from that of the reference motion, *e.g.* $S_A = \mathrm{single\_stance}$ while $S_C = \mathrm{dual\_stance}$. Such a difference is undesireable, and can occur even during small pose displacements. We define the contact state error, $E_{\mathrm{stance}}(t) \rightarrow [0, 1]$, as the ratio of time during which $S_A \neq S_C$, within a sliding window of duration $\mathrm{t_{wnd}}$.

- *Foot sliding* (E_{slide}). The feet of a simulated character sometimes slide as a result of a specific combination of internal joint torques, while the feet of the reference motion stand firmly. Since we measure body displacement relative to the COM position, this sliding often does not lead to significant errors in pose displacement. We define foot sliding error, $E_{\mathrm{slide}}(t) \rightarrow \mathbb{R}$, as the average speed in the horizontal plane of the stance foot, within a sliding window of duration $\mathrm{t_{wnd}}$. During dual stance, we use the sum of both feet.

- *Torque* (E_{torque}). Finally, we wish to control the amount of torque used by the character's actuators. We define joint torque, $E_{\text{torque}}(t) \rightarrow \mathbb{R}$, as the average summed torque of all actuated DOFs, within a sliding window of duration t_{wnd}.

The choice of window size t_{wnd} is an important component of measures E_{stance}, E_{slide}, and E_{torque}. If chosen too short, brief errors will be over-penalized, while if chosen too long, large errors can be over-compensated by other simulation segments. We use an empirically established window size of $\text{t}_{\text{wnd}} = 2\text{s}$.

Instead of constructing an optimization objective using a weighted combination of the error measures, we take on a different approach. For each of the error terms, we set a maximum acceptable threshold ($E_{\text{pose,max}}$, $E_{\text{stance,max}}$, $E_{\text{slide,max}}$, $E_{\text{torque,max}}$), and terminate the simulation if any of these maximums is exceeded (or after a predefined maximum time, t_{max}). The optimization objective is then determined using the time until termination, t_{term}. The advantage of this approach over the use of weighted terms is that it automatically minimizes wasteful computation through early termination. An additional benefit is that setting maximum acceptable thresholds is more intuitive than setting individual weights per term.

The reward function, $R(\text{P}) \rightarrow \mathbb{R}$, which we wish to maximize, is comprised of two terms. First, there is the normalized termination time, $t_{\text{term}}/t_{\text{max}}$, which represents the time a controller has been able to track the reference motion without reaching the maximum error threshold. The second term represents a 'bonus' score, based on the *normalized averages* of the error measures, $E_{\text{avg}}/E_{\text{max}}$, measured over the total time window from 0 to t_{term}. Such a bonus is useful to differentiate between trials that have similar termination times (*e.g.* in cases where a target motion has a sudden more difficult part), and to allow further optimization after $t_{\text{term}} = t_{\text{max}}$. The reward function is formulated as follows:

$$R(\text{P}) = \frac{t_{\text{term}}}{t_{\text{max}}} + \text{w}_{\text{bonus}} \frac{t_{\text{term}}}{t_{\text{max}}} \frac{1}{\|\mathbf{E}\|} \sum_{e \in \mathbf{E}} \left[1 - \frac{e_{avg}}{e_{max}} \right] \tag{6.18}$$

in which $\mathbf{E} = \left\{ E_{\text{pose}}, E_{\text{stance}}, E_{\text{slide}}, E_{\text{torque}} \right\}$ is the set of used error measures, and $\text{w}_{\text{bonus}} \in \mathbb{R}^{+}$ is a weighting term that determines the amount of bonus based on the average error (in our experiments we use $\text{w}_{\text{bonus}} = 1$). The bonus score is proportional to the normalized termination time, which is the dominant factor.

Optimization Strategy Our fitness landscape is irregular, with interdependence between parameters and many local maximums. Following recent publications in physics-based control [WFH09, WFH10, WP10, TGTL11],

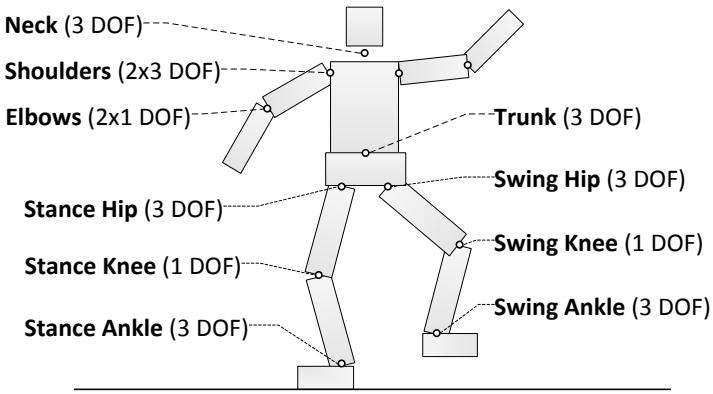


Figure 6.3: Character Degrees-of-Freedom.

Clip	Length	Description
stand	10.5s	Stand while shifting weight
arm	11.0s	Stand with rapid arm movements
wave	11.0s	Stand with waving motion
squat	9.2s	Repeated knee bends (±60 deg.)
bow	6.4s	Deep bow with arm gesture
hips	20.2s	Hula-hoop hip motion
single	9.8s	Stand on one leg, repeatedly
kick	19.7s	Single stance, fast swing forward
side	19.7s	Single stance, fast swing sideways
dance	17.0s	Dance move with alternating stance

Table 6.1: List of the motion clips used in experimentation.

we use Covariance Matrix Adaptation (CMA) [Han06] for off-line parameter optimization. CMA is an evolutionary strategy that attempts to learn the *covariance matrix* of the current region of the fitness landscape through random sampling. Many freeware implementations of CMA exist, including the Shark library [IGHM08].

6.3 Experiments

We have conducted a number of experiments to demonstrate some of the capabilities and applications of our framework. In addition to the results described in this section, we refer to the video material supplementary to this thesis.

Par	Dim	Sets	iMin	iMax	Min	Max
k_p	17	1	200	500	0	10000
k_d	17	1	20	50	0	10000
w_{cp}	1	2	300	600	-10000	10000
w_{cv}	1	2	300	600	-10000	10000
$w_t o$	1	2	100	200	-10000	10000
w_{am}	1	2	100	200	-10000	10000

Table 6.2: Overview of the 42 parameters used in off-line optimization (the sum of Dim × Sets). iMin and iMax indicate lower and upper ranges used for random initialization.

Setup For our experimentation, we have selected a set of 10 different reference motions (see Table 6.1). The motions were captured using an 8 camera Vicon system, fitting a 28 DOF character into a marker setup consisting of 41 markers (see Figure 6.3). We filtered the resulting DOF trajectories using a real-time 2nd order Butterworth filter with a cut-off frequency of 3Hz; for the 'dance' motion we use a cut-off frequency of 2Hz.

We performed physics simulation using the Open Dynamics Engine (ODE) [Smi06], using gravity constant $G = 9.81$, friction coefficient $\mu = 1.0$, and integration time step 0.0003s. We simulate springy ground contact using ODE's Error Reduction Parameter (ERP) and Constraint Force Mixing (CFM) (see [Smi06] for details). For internal joint constraints, we use ERP = 0.25, CFM = 0.0027, while for external contact constraints we use ERP = 0.0089, CFM = 0.00099. The maximum torque for each individual character joint is set to 200Nm.

Optimization For optimization, we use $E_{\text{pose,max}} = 0.1$, $E_{\text{stance,max}} = 0.5$, $E_{\text{slide,max}} = 0.25$, $E_{\text{torque,max}} = 1000$ and $t_{\text{term}} = 20$; for the 'dance' motion we use $E_{\text{stance,max}} = 1.0$ and $E_{\text{slide,max}} = 0.35$. For the CMA algorithm, we use $\lambda = 16$ and $\mu = 8$ (see [Han06] for details). See Table 6.2 for an overview of the parameters used in optimization, as well as their initialization settings and range. We declare the optimization of the control parameters a success after a threshold of $R(\mathrm{P}) = 1.8$ has been reached. The number of generations before success, as well as the estimated optimization time are displayed in Table 6.3.

External Perturbations To demonstrate the capability of our framework to respond to external perturbations, we have conducted two sets of experiments. In a first experiment, we simulate spherical objects of increasing size that are thrown towards the character. In a second experiment, we apply forces to the torso of increasing strength.

Clip	Gen	Time	Pushes	Objects
stand	7	3 min	100N	1.75kg
arm	2	1 min	180N	3.00kg
wave	2	1 min	180N	4.00kg
squat	27	16 min	100N	3.25kg
bow	37	17 min	100N	3.00kg
hips	33	28 min	120N	2.25kg
single	62	36 min	80N	3.00kg
kick	61	34 min	80N	1.25kg
side	307	274 min	80N	0.75kg
dance	600	329 min	100N	2.00kg

Table 6.3: Overview of optimization performance and resistance to external perturbations. 'Gen' is the number of generations required for optimization; 'Time' is the estimated optimization time, based on 4 × real-time performance.

In the object collision experiment, we create a sphere with density of $\rho = 100\text{kg/m}^3$, thrown at the character from random directions with a horizontal speed of 5.0m/s. The objects are created at a horizontal offset of 3.0m and a vertical offset of 1.5m of the simulated character's neck joint position. During each trial, spheres of constant weight are thrown at intervals of 1.0s. The trial is considered successful if $R(\text{P}) \geq 1.6$, after which the weight of the objects is increased by 0.25kg. The experiment is terminated after 50 generations of unsuccessful trials. The pushing force experiment is similar, but instead of objects we apply forces to the center of the trunk, for a duration of 0.2s, at an interval of 1.0s. After a successful trial, force magnitude is increased by 20N. The results are shown in Table 6.3.

Changes in Character Morphology Another advantage of physics-based character animation is the ability of controllers to adapt to changes in character morphology, while maintaining physical correctness. To demonstrate this, we optimize controllers for characters with different body dimensions, using the same set of reference motion trajectories. One tested morphology contained a short and heavy upper-body, with thin legs; another contained a long upper-body with long arms (see also Figure 6.1). In our balance control algorithm, we moved the reference target COM above the base of support and set the reference angular momentum to zero ($L_A = 0$), since the original values no longer represent physically valid motion. We have found that, with the exception of 'kick' and 'side', our control framework was able to adapt to these changes in character morphology, producing motions that fitted the new morphologies characteristically. See also the supplementary

	stand	**arm**	**wave**	**squat**	**bow**	**hips**	**single**	**kick**	**side**
stand	x	x	x		x	x			
arm	x	x	x						
wave	x	x	x		x	x			
squat	x	x	x	x	x	x			
bow	x	x	x		x				
hips	x	x	x		x	x			
single	x	x	x				x		
kick	x	x	x					x	
side	x	x	x						x

Table 6.4: Ability of controllers to be used with different motions. Crosses indicate the ability of a controller optimized for the motion on the left to be reused for the motion on top, without additional optimization.

video material.

Reusing Controllers We have also tested the capability of control parameters optimized for one motion to be used for other motions. Table 6.4 displays the results of these experiments. To increase robustness, the parameters were optimized using maximum push force perturbation.

6.4 Discussion

We have presented a framework for controlling physics-based characters using unmodified captured motion trajectories. Our control method does not require optimization of the target trajectories, nor does it require access to the equations-of-motion describing the dynamics of the character. We have demonstrated control based on motions of a diverse and dynamic nature previously unseen by this type of methods, as well as the capability to withstand external perturbations and adapt to changes in character morphology.

Based on the results, we feel there are several possible applications for our framework in production games, allowing easy integration of balanced, stylized characters that respond to external perturbations. The framework is simple enough to be implemented by a competent developer based on the descriptions provided in this chapter, using a common off-the-shelf physics engine.

Our framework relies on off-line optimization of its control parameters, and it can be argued that this is similar to requiring preprocessing of reference motion trajectories. However, as we have shown, controller settings of our framework can be reused for different motions without the need for

additional optimization. We expect that it will be possible to construct a database of parameter settings that can be used to automatically select the right set of parameters for a given character and type of motion, without the need for additional optimization.

Locomotion The results with tracking reference locomotion data are not yet as good as those of dedicated locomotion controllers. Our controller produced stiff balance corrections on heel strike, and would not retain balance longer than around 15s. A likely explanation is that our framework does not use an internal model (such as an inverted pendulum) for swing foot placement and relies on stance leg torques for balance. It is worthwhile to further investigate the possibility to extend our framework to support locomotion data, by adding swing foot balance strategies similar to those used by [YLvdP07], [CBvdP10] or [LKL10].

Effects of Individual Components We have performed some experiments to see the effects of the individual components of our balance strategy. Initial tests show that leaving out any of the components decreases performance for at least some of the reference motions. As part of future work, it is worthwhile to conduct more comprehensive experiments to gain more meaningful insights in the exact role of each component.

Torque Minimization Another direction for possible future research is the effect of torque minimization on perturbation response. A lower value of $E_{\text{torque,max}}$ (or a differently formulated error term) may promote less stiff behavior and lead to more natural compliance.

Live Data An important future direction is to investigate the possibility to use our framework with live motion data, in line with robotics research by Yamane and Hodgins [YH09]. We feel that such an approach could open up a wide range of new applications, using advanced input devices such as *Microsoft Kinect* for real-time control of an active physics-based avatar.

7

Flexible Muscle-Based Locomotion for Bipedal Creatures

Physics-based simulation is an established technique for the automatic generation of interactive natural-looking motion. To extend this approach to actively controlled virtual characters has been a longstanding research goal, in which tremendous progress has been made in recent years. Locomotion controllers have been developed that robustly deal with changes in character morphology, external perturbations and uneven terrain.

Unfortunately, in many cases the resulting motions are still not as natural as we would like. One common approach that can help improve the quality of the simulated motions is to use motion capture data as part of the control strategy. However, such methods are limited to characters and motions for which data is available. Furthermore, the biomechanical constraints that are implicit in captured motions are not preserved during the motion

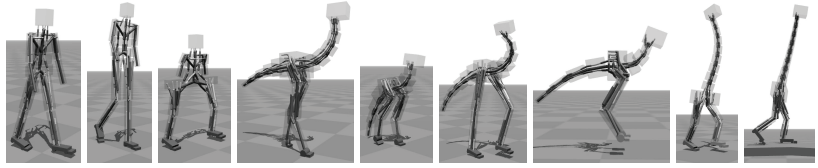


Figure 7.1: Physics-based simulation of locomotion for a variety of creatures driven by 3D muscle-based control. The synthesized controllers can locomote in real time at a range of speeds, be steered to a target heading, and can traverse variable terrain.

editing or motion retargeting that is often required to leverage limited motion data. Another approach for improving the motion quality has been to use optimization to help shape the motion, such as optimizing for minimal energy as well as task objectives. However, in the absence of biomechanical constraints, optimization objectives may lead to unnatural torque patterns or require cumbersome manual tuning. Commonly implemented joint limits and torque limits remain a crude approximation of the motion constraints that are implicit in articulated figures driven by musculotendon units.

More recently, emerging from biomechanics research, researchers have begun to develop methods that include biomechanical constraints into the simulation. Using such an approach, the natural gaits of various animals can be simulated without the need for motion data. However, the principal focus to date has been on modeling human motion, and the solutions remain limited in their locomotion abilities and robustness.

In this chapter, we make the following contributions:

- We develop a control method and optimization strategy for the simulated locomotion of fully 3D bipedal characters, including imaginary creatures, that are driven entirely by simulated muscle-based actuation. The method produces robust locomotion at given speeds to target directions and does not require pre-existing motion data. The characters can further cope with modest variations in terrain.

- We introduce muscle routing optimization as an important feature that enables and simplifies the design of muscle-based control strategies for a variety of character morphologies. Instead of needing an exact musculoskeletal model, our method requires only an approximate template of where muscles are attached and routed. The specific geometry is then optimized within the specified ranges allowed by the template, along with the parameters related to the muscle control. This approach enables the discovery of efficient muscle routings for creature models for which there exist no real-world data to draw from.

- We make use of a muscle-based approximation to Jacobian transpose control as a core component of our framework. This enables a more creature-generic and motion-generic control architecture and is applied to the majority of joints in our creature models.

7.1 Related Work

Methods for physics-based character animation that use forward dynamic simulations have been a research focus for over two decades, most often with human locomotion as the motion of interest. A survey of the considerable body of work in this area can be found in [GP12]. A rigid-link

articulated figure is typically driven by treating the joint torques at each time step as free variables, constrained by joint angle limits and joint torque limits. The underlying balance strategies commonly make direct or indirect use of foot placement, e.g., [RH91, HWBO95, LvdPF96, YLvdP07, TLC*10]. Basic locomotion capabilities have been extended in a variety of ways, including coping with terrain variations, character morphology variations, flexibly parameterized walking and running, new types of motions, new control abstractions, and methods for flexible motion sequencing. Examples here include [FvdPT01, WFH09, CBvdP09, JYL09, WFH10, WP10, MdLH10, dLMH10, YL10, CBvdP10, JL11b, LYvdPG12].

The use of motion capture data can greatly help in achieving natural physics-based locomotion. It can be used as a reference trajectory, as a well-chosen initialization point for an optimization, or as an example of an optimal solution from which to then generalize further solutions. A number of approaches exploit motion data to achieve torque-based control of simulated human motions [LHP05, SKL07, YLvdP07, dSAP08a, dSAP08b, MLPP09, LKL10, KH10, JL11b, MPP11, GPvdS12]. Many of these methods further tackle aspects of parameterization, other classes of motion, and choice of feedback abstraction.

In reality, joint torques cannot be commanded at will and must instead arise from muscles that have their own activation dynamics, force production behavior, and moment arms that change over time. They also often do not provide direct control over individual degrees of freedom, as is assumed with computed torque methods. This is because a single muscle may span multiple joints, and a single joint may be spanned by multiple muscles. Biomechanics research has developed muscle-based approaches for the simulation of a variety of human and animal motions, including lizards [ICRC07], cat hind limbs [MKT08], human jumping [PAH92], human pedaling [TAD03], and human gait [Tag95, AP01, GH10, AvdB12]. Relatedly, muscle-based simulations are being explored in the computer animation literature, where they are applied to modeling human hand motion [TSF05, SKP08], human upper body motion [LKL10], and to evaluating the realism of human motion trajectories [GvvBE10]. Most notably, the work of Geyer and Herr [GH10] has been used as the basis to animate a full 3D humanoid character by Wang et al. [WHDK12]. The motion controls are optimized with respect to an objective function that combines metabolic energy consumption and several walking-task-specific terms related to head stability and torso orientation. Together with muscle reflex models, this then produces stable forward dynamics simulations of walking at a variety of speeds that are shown to closely match human walking data.

While the majority of work on physics-based character simulation is focused on modeling human motion, control strategies have also been developed to drive simulations of block-based creatures [Sim94], swimming crea-

tures [GT95, TGTL11], walking birds or other fantastical bipeds [CBvdP09, CBvdP10, dLMH10], and quadrupeds [CKJ*11]. Physically-plausible gaits are developed *de novo*, i.e., without motion capture data, by Wampler et al. [WP09, WPP13]. Alternate kinematic approaches are developed by Hecker et al. [HRE*08] and Kry et al. [KRFC09] for producing visually plausible gaits for arbitrary creature morphologies.

Our work is most closely related to the impressive state-of-the-art work by Wang et al. [WHDK12]. We share many of the goals of this recent work, but with the following notable differences: (1) The models we develop use 3D muscles to drive all the motion of the entire body. In comparson, Wang et al. [WHDK12] use planar muscles restricted to the sagittal-plane to control the lower body, and use classic torque-based methods to control the coronal lower-body motion and the entire upper body. Our models are entirely muscle driven. (2) Our framework optimizes for the best muscle routing geometry. As such, the user only needs to provide an approximate template for muscle insertions and attachments. This is of particular utility when designing imaginary creatures or disproportional humans, for which these parameters are not known. Proper routing of 3D muscles can greatly simplify control, and we show that this aspect of the optimization significantly contributes to the ability to synthesize plausible motions. (3) We optimize for muscle physiological properties, including rest length and maximum force. These are not known a priori when designing new creatures. (4) Our control system relies heavily on target features (positions and orientations of links), and uses a muscle-based Jacobian transpose approximation to help compute target muscle activations. This contributes towards a more generic control framework. In contrast, the feedback rules of Wang et al. [WHDK12] are tailored around the human-specific reflex model developed by Geyer and Herr [GH10]. (5) Our controllers are robust enough to traverse moderate terrain variations and can perform shallow turns. (6) Most significantly, our method can be applied to automatically achieve different gaits for a variety of creatures, including imaginary creatures.

We note that the above differences are not always entirely beneficial. Because our framework targets a wider variety of creatures with a control architecture that attempts to be more generic rather than being tailored to human models, our basic approach may not achieve the same human motion fidelity as the human-specific method and results presented by Wang et al. [WHDK12]. However, researchers focusing on achieving a more faithful human gait (or any other gait) can easily extend our basic approach by adding domain-specific target features, objective terms or feedback rules.

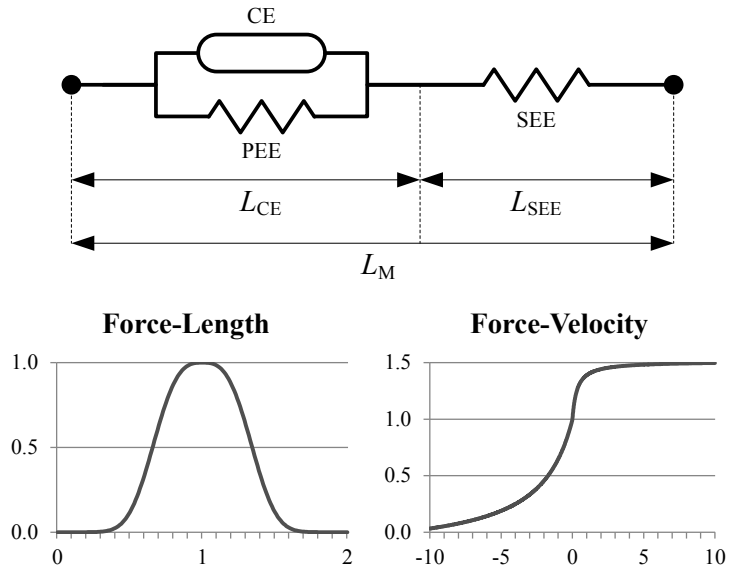


Figure 7.2: Top: The three components of a Hill-type muscle. Bottom: Normalized force-length and force-velocity relations of the contractile element.

7.2 Musculoskeletal Model

Our creature model consists of a hierarchy of rigid bodies, which are actuated using an established dynamic muscle model [GH10]. The biomechanical constraints incorporated in this model ensure the creation of realistic and smooth torque patterns. More specifically, it models the physiological properties of contractile muscle fibers and tendons (contraction dynamics), and the electro-chemical process that leads to changes in activation state (activation dynamics).

7.2.1 Muscle Contraction Dynamics

Muscles generate forces through muscle fibers contracting, based on the current activation state of the muscle. The dynamics of contraction is commonly modeled using a three element structure, also known as *Hill-type muscle model* (see Figure 7.2, top). It consists of the following elements:

- A *contractile element* (CE) that represents the muscle fibers that contract based on the muscle activation state.

- A *parallel elastic element* (PEE) that represents the passive elastic material surrounding muscle fibers.
- A *serial elastic element* (SEE) that represents the tendons that connect the muscle to the bones.

The force produced by the CE, F_{CE}, depends on the constant maximum isometrical force for the muscle, $\mathrm{F_{max}}$, the muscle activation a, fiber length L_{CE}, and contraction velocity V_{CE}:

$$F_{\mathrm{CE}} = a\ \mathrm{F_{max}}\ f_{\mathrm{L}}(L_{\mathrm{CE}})\ f_{\mathrm{V}}(V_{\mathrm{CE}}) \tag{7.1}$$

where f_{L} describes the relationship between force and length of a muscle, and f_{V} describes the relationship between force and the current contraction velocity (see Figure 7.2, bottom). Roughly speaking, a muscle can produce more force when its length is closer to its optimal length, and produces less force if it is contracting faster. The maximum isometric force $\mathrm{F_{max}}$ is a constant that we find through optimization.

The passive forces produced by the elastic elements, F_{PEE} and F_{SEE} are modeled as non-linear springs based on their length:

$$F_{\mathrm{SEE}} = f_{\mathrm{SEE}}(L_{\mathrm{M}} - L_{\mathrm{CE}}) \tag{7.2}$$

$$F_{\mathrm{PEE}} = f_{\mathrm{PEE}}(L_{\mathrm{CE}}) \tag{7.3}$$

where f_{SEE} and f_{PEE} are non-linear force-length relations and L_{M} represents the total muscle length, from which the length of the SEE can be derived. Analytical forms of f_{SEE}, f_{PEE}, f_{L} and f_{V} are described in [GSB03].

As the SEE is wired to the CE and PEE in series, the total muscle force F_{M} is subject to the force balance equation

$$F_{\mathrm{M}} = F_{\mathrm{CE}} + F_{\mathrm{PEE}} = F_{\mathrm{SEE}} \tag{7.4}$$

The length of the CE (from which the SEE length is derived) is initialized to be its optimal length, $\mathrm{L_{CE}^{opt}}$, which in combination with the tendon slack length $\mathrm{L_{SEE}^{slack}}$ defines the muscle rest length $\mathrm{L_{M}^{rest}}$:

$$\mathrm{L_{M}^{rest}} = \mathrm{L_{SEE}^{slack}} + \mathrm{L_{CE}^{opt}} \tag{7.5}$$

Both $\mathrm{L_{CE}^{opt}}$ and $\mathrm{L_{SEE}^{slack}}$ are important for the dynamic behavior of the muscle [Zaj89] and are subject to optimization. During simulation, activation state a and total muscle length L_{M} are input parameters. The first is the result of activation dynamics (Section 7.2.2), the second is determined by the geometry of the current pose (Section 7.2.3). The muscle state parameter L_{CE} is updated through integration of the contraction velocity V'_{CE}, which

is derived from Equations (7.1) and (7.4), by inverting the force-velocity relation f_V:

$$\frac{\partial L_{CE}}{\partial t} = V'_{CE} = f_V^{-1}\left[\frac{F_{SEE} - F_{PEE}}{a F_{max} f_L(L_{CE})}\right] \tag{7.6}$$

For a detailed description of this procedure we refer to Geyer and Herr [GH10].

7.2.2 Muscle Activation Dynamics

The activation state a of a muscle is altered as the result of a relatively slow electro-chemical process, based on a neural excitation signal u, which is output by the control system. This process is referred to as the *activation dynamics* and is modeled as:

$$\frac{\partial a}{\partial t} = c_a(a - u) \tag{7.7}$$

in which c_a is the constant activation and deactivation rate. In our model we use $c_a = 100s^{-1}$, following Wang et al. [WHDK12].

7.2.3 Muscle Geometry and Skeleton Interaction

The skeleton of a character and its muscles have a two-way interaction: muscles apply forces that change the skeleton pose, while the skeleton pose fully determines muscle length L_M, which then influences the contraction dynamics. The path of a muscle is determined by the locations where its tendons are attached to the bones, the bony landmarks around which the muscle wraps, and the time-varying poses of the joints that the muscle spans. In our model, we define a muscle path as a set of line segments connecting a fixed set of *via points*, which can be regarded as frictionless loopholes through which the muscle slides. This model is a simplification that has the advantage of high performance (which we require for our purpose) and omits the need to model skeleton geometry around which the muscles should wrap.

Muscle Geometry The routing of any muscle M is defined through a vector of n attachment points, $[\{b_1, \mathbf{p}_1\} \ldots, \{b_n, \mathbf{p}_n\}]$, each of which is defined by an offset point $\mathbf{p}_i$ and a body b_i to which the point is attached (see Figure 7.3). Each attachment point $\mathbf{p}_i$ is defined as a fixed offset in the coordinate frame of body b_i; it translates and rotates along with the body it is attached to. Multiple points can be attached to a single body. The first and last point, p_1 and p_n, represent the locations where the muscle tendons are attached to the skeleton, while the others are via points.

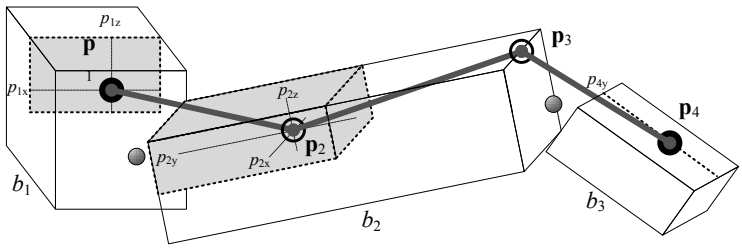


Figure 7.3: Muscle attachment points that will be optimized within a constrained region. In this example, muscle point $\mathbf{p}_1$ is constrained to a 2D surface, muscle point $\mathbf{p}_2$ is constrained to a 3D volume, $\mathbf{p}_3$ is fixed, and $\mathbf{p}_4$ is constrained to a line. The actual areas used in our experiments are shown in Figure 7.8.

The total muscle length L_{M} is equal to the summed lengths of the $n-1$ muscle segments, $[\mathbf{s}_1, \ldots, \mathbf{s}_{n-1}]$, which are found using the position of each point in the world coordinate frame, $\mathbf{p}_i^{\mathrm{W}}$:

$$L_{\mathrm{M}} = \sum_{i=1}^{n-1} \|\mathbf{s}_i\| \quad , \quad \mathbf{s}_i = \mathbf{p}_{i+1}^{\mathrm{W}} - \mathbf{p}_i^{\mathrm{W}} \tag{7.8}$$

The location of these attachment points has a great influence on both direction and magnitude of the torque a muscle can produce. The direction of the moment arm (and thereby its function) changes dynamically with the character pose; this relation is fully based on the locations of the attachment points. The amount of torque a muscle can provide depends on the projected distance between a muscle segment and the joint it spans. If it is further away, the moment arm is higher, but joint rotations will also lead to bigger changes in length, which limits the range in which the muscle can operate. Both aspects greatly affect control.

In our approach, we attempt to find efficient muscle routings through optimization. We do so by defining an area $\mathscr{P}_{\mathrm{i}}$ in which a muscle point $\mathbf{p}_{\mathrm{i}}$ must be contained. In our implementation, we allow variation of selected Cartesian components p_x, p_y or p_y within a range that is specified in an attachment template. Depending on the number of free components, the point $\mathbf{p}_{\mathrm{i}}$ is either fixed, or constrained to a line, a plane or a box (see Figure 7.3).

Force Application The total muscle length L_{M} is used in combination with the activation state a to compute the (scalar) muscle contraction force F_{M}. This force is transmitted to the skeleton at each attachment point to the body it is attached to, and generates a torque over each joint it spans. For

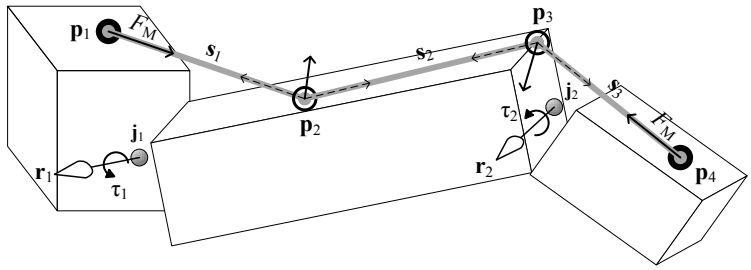


Figure 7.4: Example muscle path with four attachment points, three bodies and two joints.

each joint k, a torque τ_k is generated in the direction defined by moment arm $\mathbf{r}_k$. This moment arm corresponds to the cross-product between the direction of the muscle segment that crosses the joint, $\mathbf{s}_c$, and a vector from joint center $\mathbf{j}_k$ to any point on segment $\mathbf{s}_c$ (*e.g.* point $\mathbf{p}_c^{\mathrm{W}}$):

$$\tau_k = F_M \|\mathbf{r}_k\| \quad , \quad \mathbf{r}_k = (\mathbf{p}_c^{\mathrm{W}} - \mathbf{j}_k) \times \frac{\mathbf{s}_c}{\|\mathbf{s}_c\|} \tag{7.9}$$

The direction and size of moment arm $\mathbf{r}_k$ change as a function of the character pose, based on the geometry of $\mathbf{s}_c$ (see Figure 7.4).

7.3 Control

The goal of our muscle-based control system is to output muscle excitation signals that produce locomotion at a desired speed. An overview of our system is presented in Figure 7.5. At each step, we first update a finite state machine based on the current leg state. Next, we compose a set of target poses for a minimal set of featured body parts. These poses are based on a number of basic feedback rules for speed variation, heading control and balance. All parameters for constructing these poses are found through optimization. Finally, we compute the set of excitation signals that make muscle forces drive the featured body parts to their target positions and orientations using a muscle-based variation of Jacobian transpose control [SADM94]. We further use a small set of biologically inspired feedback rules. Time delay is incorporated in all of our excitation signals to simulate natural delay of neural systems.

7.3.1 Control States

For each leg, a separate finite state machine (FSM) keeps track of the current leg state. Similar to Wang et al. [WHDK12], we distinguish between four

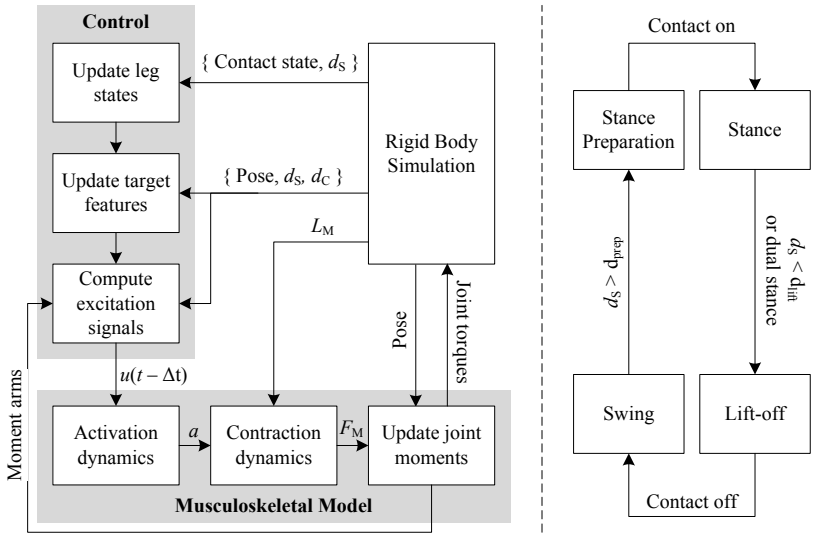


Figure 7.5: System overview. At each simulation time step, the state of each leg is updated according to the finite state machine on the right (Section 7.3.1). After that, target features are computed (Section 7.3.2) from which desired muscle excitation signals are derived. The musculoskeletal model output (Section 7.2) is fed back to the physics simulation component.

states: *stance*, *lift-off*, *swing*, and *stance preparation* (see Figure 7.5). State transitions to stance and swing occur after ground contact changes. Ground contact is measured by comparing ground reaction force to a threshold value $F_{contact}$. Lift-off is initiated for the rear leg at dual stance, or when the signed horizontal distance between foot position and the center of mass, d_S, crosses a threshold value: $d_S < d_{lift}$. Stance preparation is initiated after $d_S > d_{prep}$. $F_{contact}$, d_{lift} and d_{prep} are subject to optimization.

7.3.2 Target Features

An important part of our control system is based on target poses, which we define for a set of featured body parts. More specifically, we define a target orientation for a *trunk* body, a target position and orientation for a *head* body, and target orientations for the *leg* segments during swing and stance preparation (see Figure 7.6). The legs in our model consist of at least three segments, named *upper*, *lower*, and *foot*. All target poses are defined in a world-aligned coordinate frame, specifically one that has axes aligned with the world 'up' axis and the current facing direction of the creature (defined

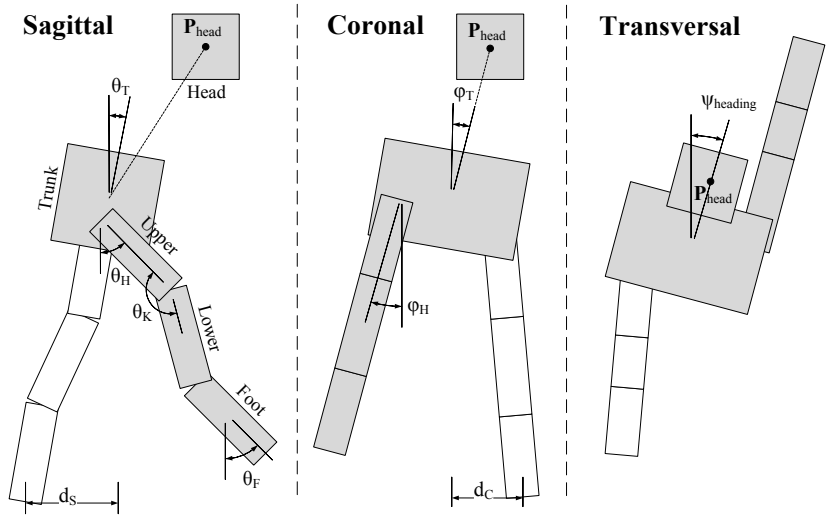


Figure 7.6: Target features for trunk, head and swing leg, shown in sagittal, coronal and transversal projection.

by the orientation of the trunk segment). The parameters to construct these poses are found during optimization.

Trunk Target The target trunk orientation, $\tilde{\mathbf{Q}}_{\text{trunk}}$ is composed of three angles, defined in the transversal, sagittal and coronal plane of the character (applied in that order). The transversal orientation is based on a target heading angle ψ_{heading}, which is a user input parameter. The sagittal orientation θ_{trunk} is based on the difference between the current center-of-mass velocity v_{COM} and desired forward velocity $\tilde{v}_{\text{forward}}$, which is also a user input parameter. The adjustment of the sagittal orientation helps the character to lean forward and backward to accelerate or decelerate. The coronal orientation φ_{trunk} is based on the difference between current transversal trunk orientation ψ_{trunk} and target heading ψ_{heading}, and helps a character lean sideways towards its target heading:

$$\theta_{\text{trunk}} = \theta^0_{\text{trunk}} + k_v(\tilde{v}_{\text{forward}} - v_{\text{COM}}) \tag{7.10}$$

$$\varphi_{\text{trunk}} = \varphi^0_{\text{trunk}} + k_h(\psi_{\text{heading}} - \psi_{\text{trunk}}) \tag{7.11}$$

The target angular velocity $\tilde{\omega}_{\text{trunk}}$ is zero.

Head Target The head segment has both a target orientation and a target position. The target orientation $\tilde{\mathbf{Q}}_{\text{head}}$ is constant in the sagittal and coronal,

and rotated to match ψ_{heading} in the transversal plane. Target position $\tilde{\mathbf{P}}_{\text{head}}$ is defined as a fixed offset in the trunk coordinate frame, and helps propagate the trunk orientation to the segments in the chain from trunk to head. The head also has a target linear velocity, $\tilde{\mathbf{v}}_{\text{head}}$, in the direction of ψ_{heading}, with a magnitude of $\tilde{v}_{\text{forward}}$. The target angular velocity $\tilde{\omega}_{\text{head}}$ is zero.

Leg Segment Targets The target upper leg orientation $\tilde{\mathbf{Q}}_{\text{upper}}$ is composed of three angles; in the transversal plane it is defined by target heading ψ_{heading}, while the orientation in the sagittal and coronal plane are based on fixed offsets and SIMBICON-style balance correction [YLvdP07]:

$$\theta_{\text{hip}} = \theta^0_{\text{hip}} + \mathrm{k}^{\theta}_{\mathrm{p}}\ d_{\mathrm{S}} + \mathrm{k}^{\theta}_{\mathrm{d}}(\tilde{v}_{\text{forward}} - v_{\mathrm{S}}) \tag{7.12}$$

$$\varphi_{\text{hip}} = \varphi^0_{\text{hip}} + \mathrm{k}^{\phi}_{\mathrm{p}}\ d_{\mathrm{C}} - \mathrm{k}^{\phi}_{\mathrm{d}} v_{\mathrm{C}} \tag{7.13}$$

in which θ^0_{hip} and ϕ^0_{hip} are offset angles, while $\mathrm{k}^{\theta}_{\mathrm{p}}$, $\mathrm{k}^{\theta}_{\mathrm{d}}$, $\mathrm{k}^{\phi}_{\mathrm{p}}$, and $\mathrm{k}^{\phi}_{\mathrm{d}}$ are control gains. The variables v_{S} and v_{C} are the current center-of-mass velocity in sagittal and coronal plane, while d_{S} and d_{C} are distances from the stance foot to the projected center-of-mass; see Figure 7.6. Initially, a rough estimate is provided for the offset angles (based on the initial pose of the character) and control parameters; their final values are found through optimization. For the sagittal parameters, different values are used during swing and stance preparation, leading to separate upper-leg targets for swing, $\tilde{\mathbf{Q}}^{\text{swing}}_{\text{upper}}$, and stance preparation, $\tilde{\mathbf{Q}}^{\text{prep}}_{\text{upper}}$. The upper leg has no target angular velocity during swing, while its target angular velocity during stance perparation, $\tilde{\omega}^{\text{prep}}_{\text{upper}}$, is zero.

The target orientation of the lower leg, $\tilde{\mathbf{Q}}_{\text{lower}}$, is defined through a fixed angle in the sagittal plane, θ_{knee}, relative to the upper leg orientation. The orientation of the foot segment, $\tilde{\mathbf{Q}}_{\text{foot}}$, is defined through a fixed angle in the sagittal plane, θ_{ankle}, relative to the ground plane. Both angles are initialized based on the model base pose, and then optimized. Both lower leg and foot have no target position or velocity.

7.3.3 Muscle-Based Feature Control

Given the set of target positions and orientations defined in the previous section, we wish to find the muscle excitation levels that cause any relevant muscle M to drive the featured bodies towards their respective targets. To accomplish this we use a muscle-based variation of Jacobian transpose control [SADM94], which attempts to find a set of muscle torques that emulate the effect of a virtual force or torque applied a specific body (see Figure 7.7). Specifically, for a body b with current state $\{\mathbf{P}_{\mathrm{b}}, \mathbf{Q}_{\mathrm{b}}, \mathbf{v}_{\mathrm{b}}, \omega_{\mathrm{b}}\}$ and target state $\{\tilde{\mathbf{P}}_{\mathrm{b}}, \tilde{\mathbf{Q}}_{\mathrm{b}}, \tilde{\mathbf{v}}_{\mathrm{b}}, \tilde{\omega}_{\mathrm{b}}\}$, we wish to minimize the state difference

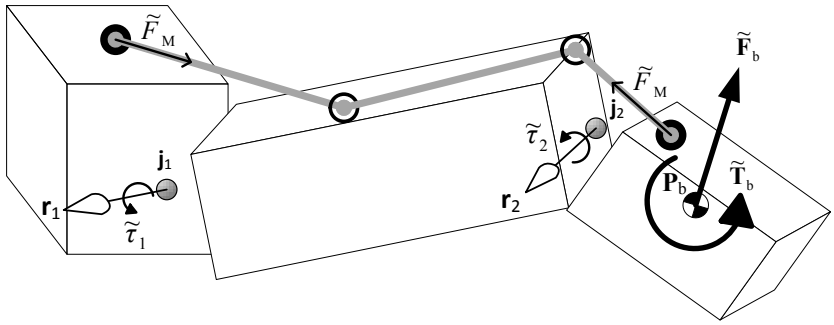


Figure 7.7: Muscle-based feature control. A virtual force and torque are applied to the rightmost limb. For each muscle, the corresponding contraction force is computed based on the muscle moment arm(s). The excitation is computed based on the difference between desired and current muscle force.

through a desired proportional-derivative (PD) feedback force $\tilde{\mathbf{F}}_b$ applied at $\mathbf{P}_b$, and a feedback torque $\tilde{\mathbf{T}}_b$:

$$\tilde{\mathbf{F}}_b = k_P\left[\tilde{\mathbf{P}}_b - \mathbf{P}_b\right] + k_v\left[\tilde{\mathbf{v}}_b - \mathbf{v}_b\right] \tag{7.14}$$

$$\tilde{\mathbf{T}}_b = k_Q\left[\mathbf{Q}_b \exp\left(\tilde{\mathbf{Q}}_b \mathbf{Q}_b^{-1}\right)\right] + k_\omega\left[\tilde{\omega}_b - \omega_b\right] \tag{7.15}$$

where $\exp(\tilde{\mathbf{Q}}_b \mathbf{Q}_b^{-1})$ is the 3D exponential map that represents the rotation from $\mathbf{Q}_b$ to $\tilde{\mathbf{Q}}_b$ (both of which are 3×3 rotation matrices). We attempt to emulate the effect of $\tilde{\mathbf{F}}_b$ and $\tilde{\mathbf{T}}_b$ through a set of torques in each joint k that is part of a chain from target body b, to a root body that is assumed stable. Each muscle M that spans over joint k produces a torque in the degree of freedom defined by moment arm $\mathbf{r}_k$; see Equation (7.9). The magnitude of this torque should match the effect of applying $\tilde{\mathbf{F}}_b$ and $\tilde{\mathbf{T}}_b$, thereby minimizing the difference between current and desired state for body b. To find this desired torque, $\tilde{\tau}_k$, we regard the rate of change of $\mathbf{P}_b$ and $\mathbf{Q}_b$, given a rotation α_k about the axis of moment arm $\mathbf{r}_k$:

$$\tilde{\tau}_k = \frac{\partial \mathbf{P}_b}{\partial \alpha_k}^T \tilde{\mathbf{F}}_b + \frac{\partial \exp(\mathbf{Q}_b)}{\partial \alpha_k}^T \tilde{\mathbf{T}}_b \tag{7.16}$$

$$= \left[\frac{\mathbf{r}_k'}{\|\mathbf{r}_k\|} \times \left[\mathbf{P}_b - \mathbf{j}_k\right]\right]^T \tilde{\mathbf{F}}_b + \frac{\mathbf{r}_k}{\|\mathbf{r}_k\|}^T \tilde{\mathbf{T}}_b \tag{7.17}$$

Based on Equation (7.9), we can derive the desired (scalar) muscle force $\tilde{F}_M$, using the magnitude of moment arm $\mathbf{r}_k$, and averaging for all m joints over which M spans:

$$\tilde{F}_M = \sum_{k=1}^{m} \frac{\tilde{\tau}_k}{m\|\mathbf{r}_k\|} \tag{7.18}$$

Note that the effect of a muscle force $\tilde{F}_{\mathrm{M}}$ only approximates the desired joint torques. The final force and torque applied to body b will in practice deviate from $\tilde{\mathbf{F}}_{\mathrm{b}}$ and $\tilde{\mathbf{T}}_{\mathrm{b}}$, and depend on the geometry and state of the individual muscles included in the chain of bodies, as well as the stability of the root body. However, we found this approximation to work well in practice, especially since the gains for $\tilde{\mathbf{F}}_{\mathrm{b}}$ and $\tilde{\mathbf{T}}_{\mathrm{b}}$ are subject to optimization.

The desired activation level $\tilde{a}_{\mathrm{M}}$ is estimated based on the maximum isometric force $\mathrm{F}_{\mathrm{max}}$:

$$\tilde{a}_{\mathrm{M}} = \frac{\tilde{F}_{\mathrm{M}}}{\mathrm{F}_{\mathrm{max}}} \tag{7.19}$$

In this estimation, we ignore the current length and velocity that are part of Equation (7.1). We have found that including length and velocity relations into our approximate inverse model for the contraction dynamics causes oscillations in muscle activation, because length and velocity change significantly over the course of neural and activation delay.

7.3.4 Muscle Activations

To compute the muscle excitation levels for our control system, we use a combination of muscle-based feature control, positive force-feedback [GSB06], and constant excitation values. We omit the use of length-based feedback rules defined in Geyer and Herr [GH10] and Wang et al. [WHDK12] in favor of our muscle-based feature control. A time delay is added to all feedback paths to simulate neural delay. For each muscle M involved in muscle-based feature control, we set the output excitation to match the desired activation:

$$\tilde{u}_{\mathrm{M}} = {\tilde{a}_{\mathrm{M}}}^{t-\Delta\mathrm{t}} \tag{7.20}$$

in which $t-\Delta\mathrm{t}$ represents the application of a delay $\Delta\mathrm{t}$. Following Geyer and Herr [GH10], we use $\Delta\mathrm{t} = 20\mathrm{ms}$ for muscles attached to the foot, $\Delta\mathrm{t} = 10\mathrm{ms}$ for muscles attached to lower leg, and $\Delta\mathrm{t} = 5\mathrm{ms}$ for muscles attached to upper leg. For other muscles, we use $\Delta\mathrm{t} = 5\mathrm{ms}$. Alternatively, the amount of delay could directly be derived from the distance of the muscle to the brain.

Trunk Orientation Feedback During stance, the orientation of the trunk is stabilized and rotated towards its target orientation $\tilde{\mathbf{Q}}_{\mathrm{trunk}}$ by all muscles connecting the trunk segment to a stance leg segment. The excitation for each HIP muscle corresponds to:

$$u_{\mathrm{HIP}}^{\mathrm{stance}} = \tilde{a}_{\mathrm{HIP}}(\tilde{\mathbf{Q}}_{\mathrm{trunk}}, \tilde{\omega}_{\mathrm{trunk}})^{t-\Delta\mathrm{t}} \tag{7.21}$$

Note that this feedback performs a rotation in 3 dimensions, depending on the planes in which the muscles operate.

Head Position and Orientation Feedback The head is moved towards its target state by all muscles connected to any body in the chain from trunk to head. The excitation is defined as:

$$u_{\mathrm{S}} = \tilde{a}_{\mathrm{S}}(\tilde{\mathbf{Q}}_{\mathrm{head}}, \tilde{\omega}_{\mathrm{head}}, \tilde{\mathbf{P}}_{\mathrm{head}}, \tilde{\mathbf{v}}_{\mathrm{head}})^{\,t-\Delta t} \tag{7.22}$$

Stance and Lift-Off Feedback For the stance leg, we do not define target orientations or positions. Instead, we rely on positive force feedback to achieve natural joint compliance [GSB06]:

$$u_{\mathrm{F+}} = \mathrm{k}_{\mathrm{M}}^{\mathrm{F+}} F_{\mathrm{M}}{}^{\,t-\Delta t} \tag{7.23}$$

in which $\mathrm{k}_{\mathrm{M}}^{\mathrm{F+}}$ is a constant feedback gain found during optimization. The length-force and velocity-force relations of any muscle ensure the excitation level does not increase indefinitely. Positive force feedback is applied to any muscle that extends the knee or ankle joint during stance and lift-off.

During lift-off, all muscles attached to the hip are fed a constant excitation of high magnitude, to initiate a leg swing. The sign of this constant depends on whether the muscle is anterior (in front), or posterior (in the back). In addition, any knee extensor muscle is fed a constant negative excitation to initiate knee swing velocity:

$$u_{\mathrm{HIP}}^{\mathrm{lift}} = \mathrm{c}_{\mathrm{HIP}}^{\mathrm{lift}} \tag{7.24}$$

$$u_{\mathrm{KNEE}}^{\mathrm{lift}} = \mathrm{c}_{\mathrm{KNEE}}^{\mathrm{lift}} \tag{7.25}$$

Swing and Stance Preparation Feedback The upper leg is guided towards its target orientation during swing and stance preparation. The upper-leg target orientation and control use separate parameters for swing and stance preparation. The lower leg is guided towards its target only during stance preparation, while the knee remains passive during swing. The ankle muscles guide the foot towards its target orientation during both swing and stance preparation. The full set of feedback rules is as follows:

$$u_{\mathrm{HIP}}^{\mathrm{swing}} = \tilde{a}_{\mathrm{HIP}}(\tilde{\mathbf{Q}}_{\mathrm{upper}}^{\mathrm{swing}})^{\,t-\Delta t} \tag{7.26}$$

$$u_{\mathrm{HIP}}^{\mathrm{prep}} = \tilde{a}_{\mathrm{HIP}}(\tilde{\mathbf{Q}}_{\mathrm{upper}}^{\mathrm{prep}}, \tilde{\omega}_{\mathrm{upper}}^{\mathrm{prep}})^{\,t-\Delta t} \tag{7.27}$$

$$u_{\mathrm{KNEE}}^{\mathrm{prep}} = \tilde{a}_{\mathrm{KNEE}}(\tilde{\mathbf{Q}}_{\mathrm{lower}})^{\,t-\Delta t} \tag{7.28}$$

$$u_{\mathrm{ANK}}^{\mathrm{swing}} = u_{\mathrm{ANK}}^{\mathrm{prep}} = \tilde{a}_{\mathrm{ANK}}(\tilde{\mathbf{Q}}_{\mathrm{foot}})^{\,t-\Delta t} \tag{7.29}$$

Constant Excitation In addition to the feedback rules stated above, all muscles have a constant excitation, which is defined separately for {stance,

Subject	Parameters	Section
Muscle physiology	3–30 *	7.2.1
Muscle geometry	12–39 *	7.2.3
State transition	3	7.3.1
Target features	14	7.3.2
Feedback control	14–63 *	7.3.3, 7.3.4
Initial character state	6	†

Table 7.1: Parameters subject to optimization. The number of parameters marked * is model dependent (see Table 7.3). † The parameters for initial character state are: initial forward lean, and initial speeds for upper swing leg, lower swing leg (and foot), upper stance leg, lower stance leg (and foot), and other bodies.

lift-off} and {swing, stance preparation}:

$$u_{\mathrm{M}}^{\mathrm{stance,lift}} = \mathrm{c}_{\mathrm{M}}^{\mathrm{stance,lift}} \tag{7.30}$$

$$u_{\mathrm{M}}^{\mathrm{swing,prep}} = \mathrm{c}_{\mathrm{M}}^{\mathrm{swing,prep}} \tag{7.31}$$

7.4 Optimization

Both our muscle model (Section 7.2) and control model (Section 7.3) introduce a large number of free parameters, which are determined through off-line optimization (see Table 7.1 for an overview). The total set of parameters, **K**, is optimized using Covariance Matrix Adaptation [Han06], with step size $\sigma = 1$ and population size $\lambda = 20$.

Objective The goal of our optimization process is to minimize the error $\bar{E}(\mathbf{K})$, which consists of the following components:

$$\bar{E}(\mathbf{K}) = \bar{E}_{\mathrm{speed}} + \bar{E}_{\mathrm{headori}} + \bar{E}_{\mathrm{headvel}} + \bar{E}_{\mathrm{slide}} + \bar{E}_{\mathrm{effort}} \tag{7.32}$$

Each right hand term is acquired by integrating a time dependent measure $E_{\mathrm{m}}(t)$ over a specific duration $\mathrm{t}_{\mathrm{max}}$:

$$\bar{E}_{\mathrm{m}} = \mathrm{W}_{\mathrm{m}} \left\{ \int_0^{\mathrm{t}_{\mathrm{max}}} E_{\mathrm{m}}(t)\, \partial\, t \right\}_{\mathrm{H}_{\mathrm{m}}} \tag{7.33}$$

in which W_{m} is measure-specific weight, while $\{\}_{\mathrm{H}_{\mathrm{m}}}$ enforces a measure-specific threshold: the value between braces is set to zero if it is lower than H_{m}. This allows for a prioritized optimization, as heavily weighted terms have greater influence until they reach their threshold. A significant difference with the error measure of Wang et al. [WHDK12] is that we

	$E^{\text{vel}}_{\text{speed}}$	$E^{\text{ori}}_{\text{head}}$	$E^{\text{vel}}_{\text{head}}$	E_{slide}	E_{effort}
Weight (W_m)	100	10	10	10	0.1
Threshold (H_m)	0.1	0.2	0.3	0.2	0

Table 7.2: Weights and thresholds for the individual error measures.

apply this threshold *after* integration, allowing incidental high values to be compensated by below-threshold averages. This is especially relevant for the initial stage of the simulation, when a character is still finding its pace. The individual weights and thresholds for each of the error terms are shown in Table 7.2.

The measure for target speed, $E_{\text{speed}}(t)$, is computed as the normalized difference between base speed and target velocity $\tilde{v}_{\text{forward}}(t)$:

$$E_{\text{speed}}(t) = \|1 - \frac{v_{base}(t)}{\tilde{v}_{\text{forward}}(t)}\| \tag{7.34}$$

in which $v_{base}(t)$ is the forward speed based on the average foot position, updated at each contact initiation. To increase head stability, we use an error measure $E_{\text{headori}}(t)$ for deviation of head orientation from its target, and $E_{\text{headvel}}(t)$ for deviation for linear head velocity from its target:

$$E_{\text{headori}}(t) = \|\mathbf{Q}^{-1}_{\text{head}}(t)\tilde{\mathbf{Q}}_{\text{head}}(t)\| \tag{7.35}$$

$$E_{\text{headvel}}(t) = \|\tilde{\mathbf{v}}_{\text{head}} - \mathbf{v}_{\text{head}}(t)\| \tag{7.36}$$

In some of our simulations, we experienced a local minimum as the result of foot sliding. Error measure $E_{\text{slide}}(t)$ prevents this by penalizing through average contact velocity $v_{\text{contact}}(t)$:

$$E_{\text{slide}}(t) = v_{\text{contact}}(t) \tag{7.37}$$

For effort minimization $E_{\text{effort}}(t)$, we use the current rate of metabolic expenditure [WHDK12].

Termination Conditions During the evaluation of $E(\mathbf{K})$, we terminate a simulation prematurely when failure is detected to save on simulation time and to help prevent local minima in the optimization. The following tests are performed during each time step:

- *Center-of-Mass Height*. To detect falling, we measure the center-of-mass position and compare its height to the initial state. The simulation is terminated if the measured height falls below a certain threshold. We use a factor of 0.9.

- *Heading*. We compare the target heading ψ_{heading} to the current trunk heading ψ_{trunk}, and terminate if they deviate over 45 degrees. In addition to keeping the character from drifting, this helps avoid a local minimum scenario where a character thrusts its feet forward during a backwards turn.

- *Self-Collision and Leg-Crossing*. We terminate on both self-collision and leg crossing, to avoid local minima where a character is unable to take another step because of self-collision. Leg-crossing occurs when the coronal left and right foot positions are reversed.

After termination, we set $E_{\text{speed}}(t) = 1$ for the remaining duration, resulting in a large penalty for failure. For the other measures we use $E_{\text{m}}(t) = 0$ for the remaining duration, to ensure their effect is minimal during the early stage of development, in which a controller is only able to take a few steps.

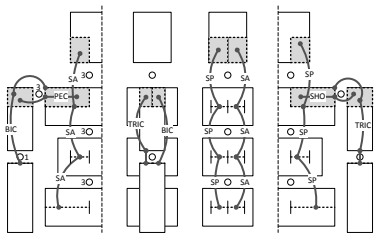
7.5 Experiments

We test our framework using Open Dynamics Engine (ODE) [Smi06], version 0.8.2, using an integration time step of 0.0003s. We use the same ground contact model as Wang et al. [WFH09], simulating a spring-damper system with $k_p = 75000$ and $k_d = 2000$ through use of ODE's CFM and ERP parameters (see [Smi06] for details), and using a friction coefficient of $\mu = 10$.

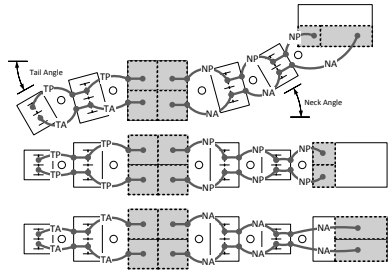
We initialize the activation level of all muscles to 0.02, and force the left leg into swing state by rotating the upper leg 20 degrees forward. Depending on the type of experiment, we evaluate for a t_{max} of either 10s or 20s. The total optimization time depends on the character model and the type of experiment; the number of evaluated generations varies between 500 and 3000. On a standard PC, optimization time takes between 2 and 12 hours. Our implementation is multi-threaded and benefits from multi-core processors. For some experiments, we use the results of earlier optimizations to speed up optimization. Aside from those cases, all parameters were initialized using a single set of values for all models and target speeds. Once optimized, a controller can drive the simulated motion in real time.

7.5.1 Creature Models

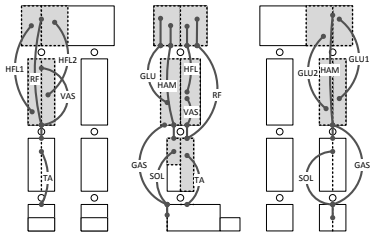
We have tested our method with four different model templates, which can be combined and parameterized to obtain a variety of biped characters as shown in Figure 7.1 and the supplementary video material. The free areas for muscle attachments, as well as the body hierarchy of each of these models



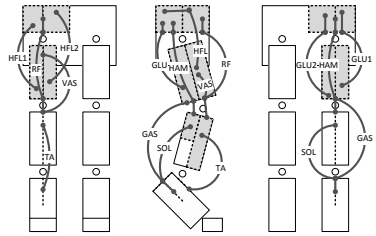
(a) Humanoid upper-body model: front, side arm, side body, and back.



(b) Neck-Tail upper-body model: side, top and bottom. The tilt angles, and number of neck and tail body segments can vary.



(c) Humanoid lower-body model: front, side and back.



(d) Inverted lower-body model: front, side and back.

Figure 7.8: Schematic overview of the permitted areas for the attachment points of each individual muscle. During optimization, each attachment point (the red dots) is allowed to move along the dotted line or area in which it is defined.

is illustrated in Figure 7.8. The elements and free parameters for each of our models are summarized in Table 7.3.

Humanoid Lower-Body Model The muscles in our humanoid lower-body model (shown in Figure 7.8(c)) largely uses the same muscles as the sagittal-plane lower-body model of Wang et al. [WHDK12], with the exception of the hip muscles. We use two out-of-plane hip flexors and two hip abductors, each of which operates simultaneously in the sagittal and coronal plane. The model has a total of 10 unique muscles for each leg; for each muscle we optimize F_{max}, L_{CE}^{opt} and L_{SEE}^{slack}. The hip and ankle joint have 3 DOFs, while the knee has 1 DOF. We include passive spring-dampers for axial hip, axial ankle

	Property	Human Legs	Human Body	Inverted Legs	Neck-Tail Body
Elements	Bodies	9	8	9	N + T + 2
	Joint DOFs	10	17	10	3N + 3T + 4
	Total muscles	20	20	20	4N + 4T + 4
	Unique muscles	10	6	10	4
Parameters	Muscle Property	30	6	30	3
	Muscle Topology	37	26	39	12
	Active Control	60	20	60	14
	Passive Control	3	1	3	0
	Total	130	53	132	29

Table 7.3: Overview of structure and parameters of our model templates.

and planar ankle rotation; the gains are subject to optimization. Following Wang et al. [WHDK12], we attach a toe segment to the foot using a joint with a spring constant of 30 Nm/rad.

Humanoid Upper-Body Model The humanoid upper-body model (shown in Figure 7.8(a)) contains 2 upper-body segments (in addition to the shared root segment), a head segment and two arm segments. The elbow joint has 1 DOF, while the shoulder, spine and neck joints have 3 DOFs. There is a total of 12 spine muscles (6 on each side) that control head position and orientation, but properties are shared for all anterior (SA) and all posterior (SP) spine muscles. The arms are controlled using 4 unique muscles per arm that roughly represent various muscle groups present in the human body. These muscles are controlled through state-dependent constant excitation only; the arm states are linked to the corresponding legs. In the upper body muscles, we optimize L_{CE}^{opt} and derive L_{SEE}^{slack} from that, because allowing rest-length optimization for spine muscles often results in stiff short muscles during optimization. We add passive spring-damper control (with optimized gains) for spine joints in the axial direction.

Inverted Lower-Body Model The inverted lower-body model (shown in Figure 7.8(d)) is similar to the humanoid lower-body, with the 'inverted knee' corresponding to the humanoid ankle. The template for muscle attachment has been modified to support long tilted feet and short upper legs, with hip joints located within the trunk body. Despite the differences in function, the initial values of the control parameters are the same as the humanoid lower-body leg model.

Neck-Tail Upper-Body Model The neck-tail upper-body model (shown in Figure 7.8(b)) supports upper bodies with a variable number of neck (N) and tail (T) segments, as well as user-defined tilt angles. Each segment is attached to its parent with a 3 DOF joint and 4 muscles. There are 4 unique muscle types: neck anterior (NA), neck posterior (NP), tail anterior (TA), and tail posterior (TP); all muscles of the same type share the same control parameters and muscle properties. The neck muscles control the target head position and orientation through muscle-based feature control (Equation (7.22)). Note that while there are individual muscles to control the motion of each neck joint, their actions are implicitly coordinated via the feature-based control strategy. The tail muscles are controlled only through constant excitation, depending on the leg state of the corresponding side. All neck and tail joints contain a low-force critically damped spring, with a spring constant of 5 Nm/rad for all DOFs.

7.5.2 Controller Capabilities

The previously described lower and upper body models can be combined by sharing the root body. Further variation can be accomplished by changing character dimensions and (for the neck-tail model) number of segments and tilt angles. We have tested a number of capabilities of our controller, for the following combinations:

- *Human*. Humanoid lower-body model with average leg lengths (upper = 0.4, lower = 0.4, foot = 0.21), and average-sized humanoid upper-body model.

- *Ostrich*. Inverted lower-body model (upper = 0.2, lower = foot = 0.45), and neck-tail upper-body model with 5 neck and 5 tail segments.

- *LongNeck*. Humanoid lower-body model with relatively short legs (upper = lower = 0.3m, foot = 0.21) and a torso of 0.5m, and neck-tail upper-body model with 8 neck segments and no tail.

- *LongLegs*. Humanoid lower-body model with long legs (upper = lower = 0.6m), same upper-body model as *Ostrich*.

Tests include locomotion at multiple speeds, steering towards target headings, robustness over uneven terrain, and robustness in the face of external perturbations. The supplementary video material displays many of these capabilities. For robustness and steering tests, we initialize controllers with results from straight walking optimizations, and declare success when a stable controller is found within 300 generations (using $t_{max} = 20$).

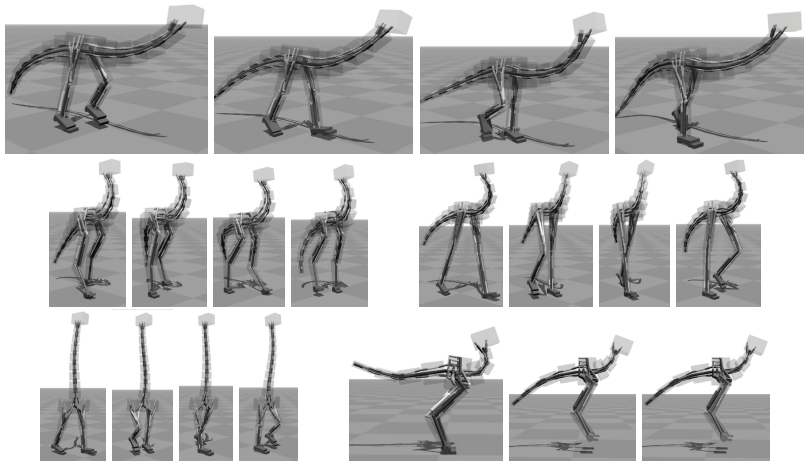


Figure 7.9: Example synthesized walking and hopping gaits.

Model	Speed	Gait	Gen	Slope	Push	Turning
Human	2.0m/s	Walk	197	±5 deg	100N	12 deg/s
Human	4.0m/s	Run	760	±3 deg	50N	12 deg/s
Ostrich	1.2m/s	Walk*	698	±5 deg	50N	6 deg/s
LongNeck	1.5m/s	Walk	541	±5 deg	100N	12 deg/s
LongLegs	2.0m/s	Walk	403	±3 deg	100N	12 deg/s

Table 7.4: Results for maximum slope, pushing force and turning speed for a selection of character models. ***Gen*** represents the number of generations before a straight walking controller reaches all of its objective function thresholds. *The ostrich model performs an 'inverted knee' stepping gait.

Speed Variation Similar to Wang et al. [WHDK12], we automatically acquire different gaits by only varying the optimization target speed. In addition to seeing running gaits at higher speeds for humanoid models, we witness a hopping style gait emerging for some creatures with inverted lower-body models, only by increasing the target velocity.

For a selection of models, Table 7.4 shows the number of generations during optimization before all error measures in the objective function reach their thresholds.

Steering In this experiment we show the ability of our controllers to make interactive turns through variation of target heading ψ_{heading}. We do so by training controllers to follow a path with random turns, with a fixed turning

velocity. The maximum turning velocities for a selection of creature models are shown in Table 7.4.

Uneven Terrain We examine the ability of our controllers to cope with uneven terrains, by placing the creature models on a 1m wide ramp, with a steepness that randomly varies each meter. Table 7.4 shows the maximum steepness ranges for a selection of creature models.

External Perturbations We examine the capability of our controllers to withstand external perturbations by applying pushes in random directions with a duration of 0.2s to the trunk body (or upper torso in case of humanoid upper-body models). Results are displayed in Table 7.4. Compared to Wang et al. [WHDK12], our controllers are somewhat less robust, but demonstrate more natural responses during lateral perturbations. This observation is in correspondence with the fact that their framework uses PD control without neural delay for coronal and upper-body balance, while our framework uses muscles with neural delay for all feedback paths.

7.5.3 Comparisons

We also examine the effect of specific components of our framework by comparing it to versions in which these components are disabled.

Disabling Biomechanical Constraints We examine the effect of the dynamic muscle model and neural delay present in our framework. More specifically, we remove the delay for our feedback signals, and omit activation and contraction dynamics by setting the output muscle force to $F_{\mathrm{M}} = u\mathrm{F}_{\mathrm{max}}$. For effort optimization, we use $E_{\mathrm{effort}}(t) = \sum F_{\mathrm{M}}$.

Together with our Jacobian transpose based control method, the altered controller becomes similar to a joint torque PD controller, with the exception that constant forces still have variable moment arms. The results in the supplementary video material clearly demonstrate increased stiffness and high-gain torque patterns in animations where biomechanical constraints are removed.

Disabling Muscle Routing Optimization We examine the effect of the optimization of muscle routing by fixing each muscle point to the center of its allowed area and removing the routing parameters from the optimization. With this restriction, optimized controllers either fail to produce robust gaits, or demonstrate clear unnatural artifacts in gait.

This effect is most apparent with some of the wider creatures (as can be seen in the supplementary video material), where the effective placement

of the optimized lateral gluteus muscles allows the character to smoothly pivot around its stance leg, thus achieving a more natural gait. In another example, which shows a humanoid with wide hips and a thin upper-body, the optimized placement of the lower abdominal muscles aids in achieving upper-body stability.

7.6 Discussion

We have introduced a flexible framework for muscle-based locomotion of bipedal creatures. The method has the versatility to support various creatures, a range of speeds, turning behavior, and robustness to external perturbations and unanticipated variations in terrain slope. Key elements include the optimization of muscle routing and the use of muscle-based approximations to Jacobian transpose control. Together, these allow for flexible and robust fully-3D muscle-driven locomotion for a variety of bipedal creatures.

The current method still has limitations. Compared to the results of Wang et al. [WHDK12], our human walking and running motions are of somewhat lesser fidelity, especially for the upper-body. This can be partially explained by the absence of target arm features in our humanoid models. We have left out such targets in favor of a generic approach, but researchers focusing on a more faithful human gait can easily reintroduce these (and other) domain-specific elements. Another important constituent is the fact that Wang et al. [WHDK12] use PD-driven torques without delay for all coronal and upper-body motion, which gives their controller exceptional lateral balance and upper-body stability. This allows their character to move more solidly in a straight line (especially during running motions), but makes it respond less naturally to perturbations. Apart from these aspects, our lower-body walking motions are very close to their state-of-the-art result. We witness a similar near-passive knee usage during swing, as well as a natural build-up of the ankle plantarflexion moment during stance. This is remarkable given the fact that we left out several of the domain-specific feedback rules introduced by Geyer and Herr [GH10].

A fundamental question shared by much of the work in this area is that of what to do when an optimization does not produce the desired results. It can be difficult to know whether to attribute the outcome to implementation errors, the optimization method finding a local minimum, the weighting of objective function terms, the given muscle routing templates, the creature morphology, or limitations of the control architecture. In practice, we have found the modular, parameterized structure of our creatures to be helpful in gaining a deeper understanding of how these various factors help shape the resulting motion patterns. The development of an improved set of authoring tools remains an important direction for future work.

Muscle-based control provides a lower-level model for generating creature motion than previous torque-based control methods, and much lower-level (more detailed) than that of kinematic models of motion. This has the potential to create significantly better models of motion, because the constraints imparted by muscle-based control now become implicit in the resulting motions. However, commensurate with this is the disadvantage of having a larger set of parameters that need to be modeled or identified from data, i.e., muscle geometry, muscle maximum forces, and other such parameters. Our results show that optimization can be used as one method to help set these extra parameters, at least for the constrained set of models and motions we have presented here. Within the scope of our framework, we demonstrate that there is a benefit to the muscle model and the muscle geometry optimization. We note, however, that this indicates that muscle models are sufficient, although they still may not always be necessary if high quality results can be achieved using other means (i.e., simpler kinematic or dynamic modeling methods). While Wang et al. [WHDK12] test and document the importance of using muscle models as compared to torque-based models, a more exact characterization of the benefits and limitations of each of these classes of models remains an important subject for future work.

Details of our simulation which could be further improved include: greater fidelity for the modeling joints such as the knees, ankles, and shoulders, more accurate muscle path wrapping models that interact with the skeleton geometry; giving further thought to the detail with which the target feature trajectories need to be modeled; the addition of anticipatory feed-forward control to the architecture; and the use of alternate dynamics simulators such as *OpenSim*, which have been thoroughly tested in the context of other biomechanics research efforts. It would be interesting to investigate the extent to which the muscle geometry optimization can be used to predict the insertion and attachment points of human musculature. An analysis of the motion with respect to the actions of antagonistic muscle pairs would also be helpful in terms of understanding the solution space. Lastly, there is a need to investigate a wider repertoire of motions, including speed transitions and more aggressive balance recovery behaviors.

8

CONCLUSION

8.1 Summary and Contributions

In this thesis we have presented the results of various studies on physics-based character animation. Our research has resulted in a number of scientific contributions, which we will summarize here.

In Chapters 2 and 3 we have presented an overview of various aspects, methods and techniques that are used in physics-based character animation research, supporting a novel classification for existing work. These chapters are intended to provide both a thorough introduction and a convenient source of reference for readers with an interest in physics-based character animation.

In Chapter 4, we have presented a method for evaluating the physical realism of kinematic animations using musculoskeletal models. We have shown how residual forces and torques can be used to estimate the ground reaction forces required for biomechanical simulation. From the resulting simulation, we have derived two novel quality measures that represent physical validity of a motion, and the validity of the amount of muscle force required to perform a motion.

In Chapter 5, we have presented a method for measuring injury levels of physics-based characters, based on data from research on human injury response levels. We have shown how to combine injury levels for individual body parts into a single measure that represents overall damage. We have found a significant correlation between our injury measure and the total injury level as perceived by human observers.

In Chapter 6, we have presented a control method for humanoid characters that is driven by unprocessed motion data; that has relatively low performance requirements; and which is relatively easy to implement and

integrate with existing physics-based simulation software. It is capable of handling motions of a diversity and agility previously unseen in comparable methods, including squatting, bowing, kicking, and dancing motions. It is also able to withstand external perturbations and adapt to changes in character morphology.

In Chapter 7, we have presented a real-time muscle-based control strategy that produces biomechanically constrained motion for a wide variety of bipedal creatures. It is the first method for this type of application in which all motion is driven entirely by simulated muscle-based actuation. We have also introduced muscle routing optimization as an important feature that enables and simplifies the design of muscle-based control strategies for a variety of morphologies. We make use of a novel muscle-based approximation to Jacobian transpose control, which contributes towards a more generic muscle-based control strategy.

8.2 General Discussion and Future Directions

In this section we will discuss some general observations and possible future directions for research on physics-based character animation. For discussions and extensions directly related to the research topics that are part of this thesis, we refer to the concluding sections of the related chapters.

8.2.1 Modeling and Optimization

The use of optimization methods has proven to be a successful strategy for automatically producing natural-looking motion [WFH09, TGTL11, WHDK12]. However, this process remains somewhat obscure, as the optimized outcome not only depends on the objective function [WHDK12], but also on the specifics of the character model [LHP05]. In the past, researchers have reported positive effects as a result of modeling soft contacts [JL11a] or biomechanical constraints [WHDK12]; but also much less evident aspects, such as subtle changes in contact modeling, have been reported to drastically influence the result of an optimized motion [dLMH10]. In our research, we have found this also to be true for aspects such as joint friction and joint limit modeling. With the introduction of more biomechanically accurate models, specifics of muscle contraction and activation dynamics models may also prove an important influence on the optimized results. Many of these factors are unexplored, and an in-depth analysis could provide a better understanding of the contribution of individual aspects of a model to the final optimized result; eventually this will permit better control over motions that are acquired through optimization.

8.2.2 Perception of Physics-Based Characters

In order to justify the use of physics-based simulation over traditional kinematics-based animation, it is important to identify to what extent users are sensitive to physically invalid motion. Even though the results are far from conclusive, evidence suggests that physical validity plays an important role in the perception of plausibility or naturalness of motion [Rei03, ODG03, HMLK10, HMO12, VHBO12]. None of these perception studies directly compare physics-based approaches to kinematics-based alternatives, though. Such a direct comparison would be precarious, because different methods can be tuned in many different ways, and data-driven approaches largely depend on the quality and pertinence of the available motion data. The choice of scenario and character model is also crucial; physics-based characters thrive during unexpected interactions, and are not limited to animations that have equivalent real-life performances. They can just as easily be used for the animation of dangerous stunts (such as diving head-first down a staircase) or for creatures that cannot be motion-captured (either because they are unwilling or unavailable). On the other hand, for scenarios in which all relevant motion data can easily be obtained, the use of physics-simulation adds much less value.

8.2.3 Perception of Failure

Physics-based characters possess a somewhat enigmatic quality that becomes evident only during failure. When a *kinematics*-based character fails, the results often look displeasing or 'fake' (such as a character penetrating a wall, or hovering above the ground), while failing *physics*-based characters typically evoke sympathy and amusement. A possible explanation is that the visual artifacts during kinematics-based failures disrupt the experience of being emerged in an alternate reality, while failing physics-based characters perhaps *asserts* the recognition of an autonomous being that struggles to adhere to the same laws of physics that we are accustomed to in our daily lives. Such a notion may carry on long beyond the failure incident, and affect the believability of subsequent non-failure animations as well. As such, it is an important topic worthy of further exploration.

One type of behavior that deserves more attention in this regard is the act of graceful falling – a largely unexplored research theme that was already briefly considered in Chapter 5. Even the most robust controller will fail to maintain balance at some point, after which a controller should take over that is able to produce natural falling behaviors. The use of a biologically-motivated injury metric could be helpful for the development of such controllers [GVE11]. In addition, the use of biomechanical constraints in a character model can also increase the naturalness of falling motions.

8.2.4 Physics-based Avatars

Some researchers have investigated the possibility of interactive control for physics-based characters [ZvdP05, WP10, MPP11, WPP13]. However, the resulting controllers seem rather unresponsive when compared to kinematics-based alternatives, which makes sense given the complex steps a physics-based character must perform to respond to user requests. As such, we do not expect physics-based controllers will replace kinematics-based approaches in applications that require responsive user-control, such as high-paced action games. However, this may only be true for traditional input devices; the current rise in novel interfaces such as the *Kinect* or the *Leap Motion* may open new possibilities [SH08, IWZL09, NWB*10]. For instance, it may be interesting to directly control a physics-based character using a full-body motion sensor, delegating part of the balance task back to the human in control. Such a scenario, were a human is an integral part of a feedback loop, could inspire exciting new applications.

8.2.5 Virtual Olympics

Finally, there is a need for proper benchmarking to allow quantitative comparison between controllers. Van de Panne [vdP00] proposes the idea of a *Virtual Olympics* event, in which physics-based characters compete with each other in various contests. Such a competition would require a detailed set of agreed-upon rules for character morphology, friction and actuation – with limitations on actuation force and energy consumption. The character models allowed in this competition could for instance be based on biomechanically accurate models [vdBGEZ*13], or models of humanoid robots such as ASIMO [CLCK05]. If such a competition would become a recurring event (similar to the *RoboCup* [KAK*97] initiative) this could bring about an enormous boost to research on physics-based character animation, biomechanics and robotics.

BIBLIOGRAPHY

[AdSP07] ABE Y., DA SILVA M., POPOVIĆ J.: Multiobjective control with frictional contacts. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2007), Eurographics Association, pp. 249–258.

[AF09] ALLEN B., FALOUTSOS P.: Evolved controllers for simulated locomotion. *Motion in Games* (2009), 219–230.

[AFP*95] AUSLANDER J., FUKUNAGA A., PARTOVI H., CHRISTENSEN J., HSU L., REISS P., SHUMAN A., MARKS J., NGO J. T.: Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Trans. on Graphics 14*, 4 (Oct. 1995), 311–336.

[AG85] ARMSTRONG W., GREEN M.: The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer 1*, 4 (1985), 231–240.

[Ale01] ALEXANDER R. M.: Design by numbers. *Nature 412*, 6847 (Aug. 2001), 591.

[AP93] ANDERSON F. C., PANDY M. G.: Storage and utilization of elastic strain energy during jumping. *Journal of biomechanics 26*, 12 (Dec. 1993), 1413–27.

[AP01] ANDERSON F., PANDY M.: Dynamic optimization of human walking. *Journal of Biomechanical Eng. 123* (2001), 381.

[AP06] ABE Y., POPOVIĆ J.: Interactive animation of dynamic manipulation. *Proc. of the 2006 ACM SIGGRAPH/Eurographics symp. on Computer animation* (2006), 195–204.

[AvdB12] ACKERMANN M., VAN DEN BOGERT A. J.: Predictive simulation of gait at low gravity reveals skipping as the preferred locomotion strategy. *Journal of Biomechanics 45*, 7 (2012), 1293–8.

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. In *Proc. of the int. conf. on Computer Graphics and interactive techniques in Australia and Southeast Asia* (2007), pp. 281 – 288.

[Bis94] BISHOP C.: Neural networks and their applications. *Review of Scientific Instruments 65*, 6 (1994), 1803–1832.

[BOHL74] BAKER S., O'NEILL B., HADDON JR W., LONG W.: The injury severity score: a method for describing patients with multiple injuries and evaluating emergency care. *The Journal of trauma 14*, 3 (1974), 187.

[BT08] BYL K., TEDRAKE R.: Approximate optimal control of the compass gait on rough terrain. *Int. Conf. on Robotics and Automation* (2008), 1258–1263.

[CBvdP09] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust Task-based Control Policies for Physics-based Characters. *ACM Trans. on Graphics 28*, 5 (2009).

[CBvdP10] COROS S., BEAUDOIN P., VAN DE PANNE M.: Generalized biped walking control. *ACM Trans. on Graphics 29*, 4 (2010).

[CBYvdP08] COROS S., BEAUDOIN P., YIN K. K., VAN DE PANNE M.: Synthesis of constrained walking skills. *ACM Trans. on Graphics 27*, 5 (Dec. 2008).

[Che07] CHESTNUTT J.: Navigation planning for legged robots. *PhD Thesis, Carnegie Mellon University*, December (2007).

[CKJ*11] COROS S., KARPATHY A., JONES B., REVERET L., VAN DE PANNE M.: Locomotion skills for simulated quadrupeds. *ACM Trans. on Graphics 30*, 4 (2011).

[CKK*98] CRANDALL J., KUPPA S., KLOPP G., HALL G., PILKEY W., HURWITZ S.: Injury mechanisms and criteria for the human foot and ankle under axial impacts to the foot. *International Journal of Crashworthiness 3*, 2 (1998), 147–162.

[CLCK05] CHESTNUTT J., LAU M., CHEUNG G., KUFFNER J.: Footstep planning for the Honda ASIMO humanoid. *Proc. of the IEEE Int. Conf. on Robotics and Automation 1* (2005), 629–634.

[Cou01] COUTINHO M.: *Dynamic simulations of multibody systems*. Springer Verlag, 2001.

[CRTW05] COLLINS S., RUINA A., TEDRAKE R., WISSE M.: Efficient bipedal robots based on passive-dynamic walkers. *Science (New York, N.Y.) 307*, 5712 (2005), 1082–5.

[DAA*07] DELP S. L., ANDERSON F. C., ARNOLD A. S., LOAN P., HABIB A., JOHN C. T., GUENDELMAN E., THELEN D. G.: OpenSim: open-source software to create and analyze dynamic simulations of movement. *IEEE trans. on bio-medical eng. 54*, 11 (2007), 1940–50.

[dL96] DE LEVA P.: Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *Journal of biomechanics 29*, 9 (1996), 1223–1230.

[dLH09] DE LASA M., HERTZMANN A.: Prioritized optimization for task-space control. *2009 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 3* (Oct. 2009), 5755–5762.

[dLMH10] DE LASA M., MORDATCH I., HERTZMANN A.: Feature-Based Locomotion Controllers. *ACM Trans. on Graphics 29*, 3 (2010).

[dSAP08a] DA SILVA M., ABE Y., POPOVIĆ J.: Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (Aug. 2008), 1–10.

[dSAP08b] DA SILVA M., ABE Y., POPOVIC J.: Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum 27*, 2 (2008), 371–380.

[dSDP09] DA SILVA M., DURAND F., POPOVIĆ J.: Linear Bellman combination for control of character animation. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (July 2009), 1–10.

[EMHvdB07] ERDEMIR A., MCLEAN S., HERZOG W., VAN DEN BOGERT A.: Model-based estimation of muscle forces exerted during movements. *Clinical Biomechanics 22*, 2 (2007), 131–154.

[EMO10] ENNIS C., MCDONNELL R., O'SULLIVAN C.: Seeing is believing: Body motion dominates in multisensory conversations. *ACM Trans. on Graphics 29*, 4 (2010).

[Fal01] FALOUTSOS P.: The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics 25*, 6 (Dec. 2001), 933–953.

[FDCM97] FAURE F., DEBUNNE G., CANI M., MULTON F.: Dynamic analysis of human walking. In *Computer animation and simulation* (1997), pp. 53–65.

[Fea08] FEATHERSTONE R.: *Rigid body dynamics algorithms*. Springer-Verlag New York Inc, 2008.

[FKW77] FOSTER J., KORTGE J., WOLANIN M.: *Hybrid III-a biomechanically-based crash test dummy*. Tech. rep., Society of Automotive Engineers, 400 Commonwealth Dr, Warrendale, PA, 15096, USA,, 1977.

[FP03] FANG A. C., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Trans. on Graphics 22*, 3 (July 2003), 417.

[Fro79] FROHLICH C.: Do springboard divers violate angular momentum conservation? *American Journal of Physics 47*, 0002-9505 (1979), 583–592.

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *ACM SIGGRAPH Papers* (2001), pp. 251–260.

[Gar90] GARIS H. D.: Genetic programming: Building artificial nervous systems with genetically programmed neural network modules. *Proc of the 7th Int. Conf. on Machine Learning* (1990), 207.

[GH10] GEYER H., HERR H.: A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. *IEEE transactions on neural systems and rehabilitation engineering 18*, 3 (June 2010), 263–73.

[GK04] GOSWAMI A., KALLEM V.: Rate of change of angular momentum and balance maintenance of biped robots. In *IEEE International Conference on Robotics and Automation* (2004), vol. 4, pp. 3785–3790.

[GP12] GEIJTENBEEK T., PRONOST N.: Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. *Computer Graphics Forum 31*, 8 (Dec. 2012), 2492–2515.

[GPEO11] GEIJTENBEEK T., PRONOST N., EGGES A., OVERMARS M. H.: Interactive character animation using simulated physics. *Eurographics-State of the Art Reports 2* (2011).

[GPvdS12] GEIJTENBEEK T., PRONOST N., VAN DER STAPPEN A.: Simple Data-Driven Control for Simulated Bipeds. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (Lausanne, Switzerland, 2012), Kry P., Lee J., (Eds.), The Eurographics Association, pp. 211–219.

[Gra98] GRASSIA F. S.: Practical Parameterization of Rotations Using the Exponential Map. *The Journal of Graphics Tools 3*, 3 (1998), 1–13.

[GSB03] GEYER H., SEYFARTH A., BLICKHAN R.: Positive force feedback in bouncing gaits? *Proc. of the Royal Society of London. Series B: Biological Sciences 270*, 1529 (2003), 2173–2183.

[GSB06] GEYER H., SEYFARTH A., BLICKHAN R.: Compliant leg behaviour explains basic dynamics of walking and running. *Proc. of Biological sciences / The Royal Society 273*, 1603 (Nov. 2006), 2861–7.

[GT95] GRZESZCZUK R., TERZOPOULOS D.: Automated learning of muscle-actuated locomotion through control abstraction. In *ACM SIGGRAPH Papers* (1995), pp. 63–70.

[GTH98] GRZESZCZUK R., TERZOPOULOS D., HINTON G.: Neuroanimator: Fast neural network emulation and control of physics-based models. In *ACM SIGGRAPH Papers* (1998), pp. 9–20.

[GvdBvBE10] GEIJTENBEEK T., VAN DEN BOGERT A., VAN BASTEN B., EGGES A.: Evaluating the physical realism of character animations using musculoskeletal models. In *Motion in Games* (Zeist, The Netherlands, 2010), Boulic R., Chrysantou Y., Komura T., (Eds.), vol. 6459, Springer-Verlag Berlin Heidelberg, pp. 11–22.

[GVE11] GEIJTENBEEK T., VASILESCU D., EGGES A.: Injury Assessment for Physics-Based Characters. In *Motion in Games* (Edinburgh, UK, 2011), Allbeck J., Faloutsos P., (Eds.), vol. 7060, Springer-Verlag Berlin Heidelberg, pp. 74–85.

[GvvBE10] GEIJTENBEEK T., VAN DEN BOGERT A. J., VAN BASTEN B. J. H., EGGES A.: Evaluating the physical realism of character animations using musculoskeletal models. In *Motion in Games*. Springer, 2010, pp. 11–22.

[GW06] GENNARELLI T. A., WODZIN E.: AIS 2005: a contemporary injury scale. *Injury 37*, 12 (Dec. 2006), 1083–91.

[GY11] GIOVANNI S., YIN K.: LocoTest : Deploying and Evaluating Physics-based Locomotion on Multiple Simulation Platforms. In *Motion in Games* (2011), Springer, pp. 227–241.

[Han06] HANSEN N.: The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation* (2006), 75–102.

[HKL98] HUTCHINSON J., KAISER M. J., LANKARANI H. M.: The Head Injury Criterion (HIC) functional. *Applied Mathematics and Computation 96*, 1 (Oct. 1998), 1–16.

[HMLK10] HOYET L., MULTON F., LECUYER A., KOMURA T.: Can we distinguish biological motions of virtual humans?: perceptual study with captured motions of weight lifting. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology* (2010), ACM, pp. 87–90.

[HMO12] HOYET L., MCDONNELL R., O'SULLIVAN C.: Push it real. *ACM Transactions on Graphics 31*, 4 (July 2012), 1–9.

[HMOA03] HASE K., MIYASHITA K., OK S., ARAKAWA Y.: Human gait simulation with a neuromusculoskeletal model and evolutionary computation. *The Journal of Visualization and Computer Animation 14*, 2 (2003), 73–92.

[HMPH05] HOFMANN A., MASSAQUOI S., POPOVIC M., HERR H.: A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. In *Proc. of the Int. Conf. on Intelligent Robots and Systems* (2005), vol. 2, IEEE, pp. 1952–1959.

[Hod91] HODGINS J.: Biped gait transitions. In *Proceedings of the IEEE International Conference on Robotics and Automation* (1991), pp. 2092–2097.

[Hog90] HOGAN N.: Mechanical Impedance of Single-and Multi-Articular Systems. *Multiple muscle systems: biomechanics and movement organization* (1990), 149.

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated behaviors for new characters. In *ACM SIGGRAPH Papers* (1997), pp. 153–162.

[HRE*08] HECKER C., RAABE B., ENSLOW R. W., DEWEESE J., MAYNARD J., VAN PROOIJEN K.: Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. on Graphics 27*, 3 (Aug. 2008), 1.

[HRS94] HOLLARS M., ROSENTHAL D., SHERMAN M.: SD/Fast user's manual. *Symbolic Dynamics Inc* (1994).

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *ACM SIGGRAPH Papers* (1995), pp. 71–78.

[HZ11] HERTZMANN A., ZORDAN V.: Physics-Based Characters. *IEEE Computer Graphics and Applications 31*, 4 (2011), 20–21.

[IAF07] IKEMOTO L., ARIKAN O., FORSYTH D.: Quick transitions with cached multi-way blends. In *Proc. of the 2007 symp. on interactive 3D graphics and games* (2007), ACM, pp. 145–151.

[ICRC07] IJSPEERT A. J., CRESPI A., RYCZKO D., CABELGUEN J.-M.: From swimming to walking with a salamander robot driven by a spinal cord model. *Science (New York, N.Y.) 315*, 5817 (Mar. 2007), 1416–20.

[IGHM08] IGEL C., GLASMACHERS T., HEIDRICH-MEISNER V.: Shark. *Journal of Machine Learning Research 9* (2008), 993–996.

[Isa87] ISAACS P.: Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics, ACM SIGGRAPH Proceedings 21*, 4 (1987), 215–224.

[IWZL09] ISHIGAKI S., WHITE T., ZORDAN V. B., LIU C. K.: Performance-based control interface for character animation. *ACM Trans. on Graphics 28*, 3 (July 2009).

[Jaz10] JAZAR R.: *Theory of applied robotics: kinematics, dynamics, and control*. Springer Verlag, 2010.

[JL11a] JAIN S., LIU C.: Controlling Physics-Based Characters Using Soft Contacts. *ACM Trans. on Graphics 30*, 6 (2011).

[JL11b] JAIN S., LIU C.: Modal-space control for articulated characters. *ACM Trans. on Graphics 30*, 5 (2011).

[JWdC89] JANSSEN E. G., WISMANS J., DE COO P. J. A.: Comparison of Eurosid and Cadaver Responses in Side Impact. In *Twelfth International Technical Conference on Experimental Safety Vehicles* (1989).

[JYL09] JAIN S., YE Y., LIU C. K.: Optimization-based interactive motion synthesis. *ACM Trans. on Graphics 28*, 1 (2009).

[KAK*97] KITANO H., ASADA M., KUNIYOSHI Y., NODA I., OSAWA E., MATSUBARA H.: RoboCup: A challenge problem for AI. *AI magazine 18*, 1 (1997), 73.

[KB96] KO H., BADLER N.: Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications* (1996), 50–59.

[KBF*03] KERRIGAN J., BHALLA K., FUNK J., MADELEY N., BOSE D.: Experiments for Establishing Pedestrian-Impact Lower Limb Injury Criteria. *SAE Technical Paper* (2003).

[KH10] KWON T., HODGINS J.: Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2010), pp. 129–138.

[Kha87] KHATIB O.: A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation 3*, 1 (1987), 43–53.

[KKI06] KUDOH S., KOMURA T., IKEUCHI K.: Stepping motion for a human-like character to maintain balance against large perturbations. *IEEE Int. Conf. Robotics and Automation* (2006).

[KKK*03a] KAJITA S., KANEHIRO F., KANEKO K., FUJIWARA K., HARADA K., YOKOI K., HIRUKAWA H.: Biped walking pattern generation by using preview control of zero-moment point. In *IEEE Int. Conf. on Robotics and Automation* (2003), vol. 2, pp. 1620–1626.

[KKK*03b] KAJITA S., KANEHIRO F., KANEKO K., FUJIWARA K., YOKOI K., HIRUKAWA H.: Biped walking pattern generation by a simple three-dimensional inverted pendulum model. *Advanced Robotics 17*, 2 (2003), 131–147.

[KL96] KANE T., LEVINSON D.: Dynamics online: theory and implementation with AUTOLEV. *Online Dynamics, Inc* (1996).

[KLM96] KAELBLING L., LITTMAN M., MOORE A.: Reinforcement learning: A survey. *Journal of Artificial Intelligence Research 4* (1996), 237–285.

[KP11] KIM J., POLLARD N.: Direct Control of Simulated Nonhuman Characters. *Computer Graphics and Applications, IEEE 31*, 4 (2011), 56–65.

[KRFC09] KRY P., REVERET L., FAURE F., CANI M.-P.: Modal Locomotion: Animating Virtual Characters with Natural Vibrations. *Computer Graphics Forum 28*, 2 (Apr. 2009), 289–298.

[KSnL05] KELLY R., SANTIBÁÑEZ V., LORIA A.: *Control of Robot Manipulators in Joint Space*. Springer-Verlag, 2005.

[LHP05] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (2005), 1071.

[LHP06] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Composition of complex optimal multi-character motions. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2006), pp. 215–222.

[Lie77] LIEGEOIS A.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. On Systems Man And Cybernetics 7*, 12 (1977), 868–871.

[LKL10] LEE Y., KIM S., LEE J.: Data-driven biped control. *ACM Trans. on Graphics 29*, 4 (July 2010), 129.

[LST09] LEE S., SIFAKIS E., TERZOPOULOS D.: Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. on Graphics 28*, 4 (Aug. 2009), 99.

[LvdPF96] LASZLO J., VAN DE PANNE M., FIUME E.: Limit cycle control and its application to the animation of balancing and walking. In *ACM SIGGRAPH Papers* (1996), pp. 155–162.

[LvdPF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. In *ACM SIGGRAPH Papers* (2000), pp. 201–208.

[LWZB90] LEE P., WEI S., ZHAO J., BADLER N. I.: Strength guided motion. *ACM SIGGRAPH Computer Graphics 24*, 4 (Sept. 1990), 253–262.

[LYvdP*10] LIU L., YIN K., VAN DE PANNE M., SHAO T., XU W.: Sampling-based contact-rich motion control. *ACM Trans. on Graphics 29* (July 2010).

[LYvdPG12] LIU L., YIN K., VAN DE PANNE M., GUO B.: Terrain Runner: Control, Parameterization, Composition, and Planning for Highly Dynamic Motions. *ACM Trans. on Graphics 31*, 6 (Nov. 2012), 1.

[MAEC07] MORIMOTO J., ATKESON C. G., ENDO G., CHENG G.: Improving humanoid locomotive performance with learnt approximated dynamics via Gaussian processes for regression. *Int. Conf. on Intelligent Robots and Systems* (Oct. 2007), 4234–4240.

[McG90] MCGEER T.: Passive dynamic walking. *The International Journal of Robotics Research 9*, 2 (Jan. 1990), 62.

[MdLH10] MORDATCH I., DE LASA M., HERTZMANN A.: Robust Physics-Based Locomotion Using Low-Dimensional Planning. *ACM Trans. on Graphics 29*, 4 (2010).

[Mer84] MERTZ H.: Injury assessment values used to evaluate Hybrid III response measurements. *NHTSA docket 74* (1984), 14.

[MKT08] MAUFROY C., KIMURA H., TAKASE K.: Towards a general neural controller for quadrupedal locomotion. *Neural networks : the official journal of the International Neural Network Society 21*, 4 (May 2008), 667–81.

[MLPP09] MUICO U., LEE Y., POPOVIĆ J., POPOVIĆ Z.: Contact-aware nonlinear control of dynamic characters. *ACM Trans. on Graphics 28*, 3 (July 2009).

[MME86] MORGAN R., MARCUS J., EPPINGER R.: Side impact-the biofidelity of NHTSA's proposed ATD and efficacy of TTI. *SAE Tech. Pap. Ser* (1986).

[MPI97] MERTZ H., PRASAD P., IRWIN A.: *Injury risk curves for children and adults in frontal and rear collisions*. Tech. rep., Soc. of Automotive Engineers, 1997.

[MPP11] MUICO U., POPOVIĆ J., POPOVIĆ Z.: Composite control of physically simulated characters. *ACM Trans. on Graphics 30*, 3 (May 2011).

[MTK98] MIYAKOSHI S., TAGA G., KUNIYOSHI Y.: Three dimensional bipedal stepping motion using neural oscillators-towards humanoid motion in the real world. *Int. Conf. on Intelligent Robots and Systems 18*, 1 (1998), 87–97.

[MTP12] MORDATCH I., TODOROV E., POPOVIĆ Z.: Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. on Graphics 31*, 4 (July 2012), 1–8.

[MV98] MICHAELS C. F., VRIES M. M. D.: Higher Order and Lower Order Variables in the Visual Perception of Relative Pulling Force. *Journal of Experimental Psychology: Human Perception and Performance 24*, 2 (1998), 20.

[MWB03] MULLER M. E., WEBBER C. E., BOUXSEIN M. L.: Predicting the failure load of the distal radius. *Osteoporosis International 14*, 4 (June 2003), 345–52.

[MY11] MURAI A., YAMANE K.: A neuromuscular locomotion controller that realizes human-like responses to unexpected disturbances. *International Conference on Robotics and Automation (ICRA) 1*, 2-3 (2011), 1997–2002.

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C. R.: Momentum control for balance. *ACM Trans. on Graphics 28*, 3 (2009).

[MZW99] MATARIĆ M., ZORDAN V., WILLIAMSON M.: Making complex articulated agents dance. *Autonomous Agents and Multi-Agent Systems 2*, 1 (1999), 23–43.

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2002), ACM, pp. 81–88.

[NHR99] NG A. Y., HARADA D., RUSSELL S.: Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. of the 16th Int. Conf. on Machine Learning* (1999), pp. 278–287.

[NM93] NGO J., MARKS J.: Physically realistic motion synthesis in animation. *Evolutionary Computation 1*, 3 (Sept. 1993), 235–268.

[NVCNZ08] NUNES R. F., VIDAL C. A., CAVALCANTE-NETO J., ZORDAN V. B.: Simple Feedforward Control for Responsive Motion

Capture-Driven Simulations. *Advances in Visual Computing* (2008), 488–497.

[NWB*10] NGUYEN N., WHEATLAND N., BROWN D., PARISE B., LIU C. K., ZORDAN V.: Performance capture with physical interaction. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (July 2010), pp. 189–195.

[ODG03] O'SULLIVAN C., DINGLIANA J., GIANG T.: Evaluating the visual fidelity of physically based animations. *ACM Trans. on Graphics 22*, 3 (2003), 527–536.

[OM01] OSHITA M., MAKINOUCHI A.: A dynamic motion control technique for human-like articulated figures. *Computer Graphics Forum 20*, 3 (2001), 192–203.

[Ott03] OTTEN E.: Inverse and forward dynamics: models of multibody systems. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences 358*, 1437 (Sept. 2003), 1493–1500.

[PAH92] PANDY M., ANDERSON F., HULL D.: A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Journal of Biomechanical Engineering, Transactions of the ASME 114*, 4 (1992), 450–460.

[PCTD01] PRATT J., CHEW C., TORRES A., DILWORTH P.: Virtual model control: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research 20*, 2 (Feb. 2001), 129–143.

[Pet81] PETRUCELLI E.: The abbreviated injury scale: Evolution, usage and future adaptability. *Accident Analysis & Prevention 13*, 1 (Mar. 1981), 29–35.

[PLF*00] POLLACK J., LIPSON H., FICICI S., FUNES P., HORNBY G., WATSON R.: Evolutionary techniques in physical robotics. In *Evolvable systems: from biology to hardware: third int. conf.* (2000), Springer Verlag, p. 175.

[PMO10] PRAŽÁK M., MCDONNELL R., O'SULLIVAN C.: Perceptual evaluation of human animation timewarping. In *ACM SIGGRAPH ASIA 2010 Sketches* (2010), ACM, p. 30.

[PP10] PEJSA T., PANDZIC I.: State of the Art in Example-Based Motion Synthesis for Virtual Characters in Interactive Applications. *Computer Graphics Forum 29*, 1 (2010), 202–226.

[PW99] POPOVIĆ Z., WITKIN A.: Physically based motion transformation. In *ACM SIGGRAPH Papers* (1999), pp. 11–20.

[PZ05] POLLARD N. S., ZORDAN V. B.: Physically based grasping control from example. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2005), ACM Press, p. 311.

[RDS*03] RASMUSSEN J., DAMSGAARD M., SURMA E., CHRISTENSEN S., DE ZEE M., VONDRAK V.: Anybody-a software system for ergonomic optimization. In *Fifth World Congress on Structural and Multidisciplinary Optimization* (2003).

[Rei03] REITSMA P.: Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM SIGGRAPH 2003 Papers 22*, 3 (2003), 537–542.

[RF81] RUNESON S., FRYKHOLM G.: Visual perception of lifted weight. *Journal of exp. psychology: Human perception and performance 7*, 4 (Aug. 1981), 733–40.

[RH91] RAIBERT M. H., HODGINS J. K.: Animation of dynamic legged locomotion. *ACM SIGGRAPH Computer Graphics 25*, 4 (July 1991), 349–358.

[RH02] REIL T., HUSBANDS P.: Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation 6*, 2 (Apr. 2002), 159–168.

[RM01] REIL T., MASSEY C.: Biologically inspired control of physically simulated bipeds. *Theory in Biosciences 120*, 3-4 (Dec. 2001), 327–339.

[RPE05] REN L., PATRICK A., EFROS A.: A data-driven approach to quantifying natural human motion. *ACM Trans. on Graphics 24*, 3 (2005), 1090–1097.

[SADM94] SUNADA C., ARGAEZ D., DUBOWSKY S., MAVROIDIS C.: A coordinated Jacobian transpose control for mobile multilimbed robotic systems. In *IEEE Int. Conf. on Robotics and Automation* (1994), pp. 1910–1915.

[SC92a] STEWART A., CREMER J.: Animation of 3d human locomotion: Climbing stairs and descending stairs. In *3rd Eurographics workshop on Animation* (1992), pp. 1–18.

[SC92b] STEWART A., CREMER J.: Beyond keyframing: An algorithmic approach to animation. In *Graphics Interface* (1992), pp. 273–281.

[SCAF07] SHAPIRO A., CHU D., ALLEN B., FALOUTSOS P.: A dynamic controller toolkit. In *Proceedings of the 2007 ACM SIGGRAPH symposium on Video games* (2007), ACM, p. 20.

[SFNTH05] SHAPIRO A., FALOUTSOS P., NG-THOW-HING V.: Dynamic Animation and Control Environment. In *Proceedings of Graphics Interface 2005* (May 2005), pp. 61–70.

[SH05] SAFONOVA A., HODGINS J. K.: Analyzing the physical correctness of interpolated human motion. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2005), ACM Press, pp. 171–180.

[SH08] SHIRATORI T., HODGINS J.: Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Trans. on Graphics 27*, 5 (2008), 123.

[SHP04] SAFONOVA A., HODGINS J., POLLARD N.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH Papers* (2004), pp. 514–521.

[Sim94] SIMS K.: Evolving virtual creatures. In *ACM SIGGRAPH Papers* (1994), pp. 15–22.

[SKG03] SHIN H. J., KOVAR L., GLEICHER M.: Physical touch-up of human motions. *Proc. of the 11th Pacific Conf. on Comp. Graphics and Appl.* (2003), 194–203.

[SKL07] SOK K., KIM M., LEE J.: Simulating biped behaviors from human motion data. *ACM Trans. on Graphics 26*, 3 (2007), 107.

[SKP08] SUEDA S., KAUFMAN A., PAI D. K.: Musculotendon simulation for hand animation. *ACM Trans. on Graphics 27*, 3 (2008), 83.

[Smi06] SMITH R.: Open Dynamics Engine User Guide v0.5, 2006.

[SPF03] SHAPIRO A., PIGHIN F., FALOUTSOS P.: Hybrid control for interactive character animation. *Pacific Graphics* (2003), 455–461.

[SSGL*01] STEVENSON M., SEGUI-GOMEZ M., LESCOHIER I., DI SCALA C., MCDONALD-SMITH G.: An overview of the injury severity score and the new injury severity score. *Injury Prevention 7*, 1 (Mar. 2001), 10–3.

[Sta04] STANLEY K.: *Efficient evolution of neural networks through complexification*. The University of Texas at Austin, 2004.

[SvdP05] SHARON D., VAN DE PANNE M.: Synthesis of controllers for stylized planar bipedal walking. *Proc. of the Int. Conf. on Robotics and Automation* (2005), 2387–2392.

[TAD03] THELEN D., ANDERSON F., DELP S.: Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics 36* (2003), 321–328.

[Tag95] TAGA G.: A model of the neuro-musculo-skeletal system for human locomotion. *Biological Cybernetics 73*, 2 (1995), 97–111.

[Tag98] TAGA G.: A model of the neuro-musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance. *Biological Cybernetics 78*, 1 (1998), 9–17.

[TGTL11] TAN J., GU Y., TURK G., LIU C.: Articulated swimming creatures. *ACM Trans. on Graphics 30*, 4 (2011), 58.

[TLC*09] TSAI Y., LIN W., CHENG K., LEE J., LEE T.: Real-Time Physics-Based 3D Biped Character Animation Using an Inverted Pendulum Model. *IEEE Transactions on Visualization and Computer Graphics 16*, 2 (2009), 325–337.

[TLC*10] TSAI Y.-Y., LIN W.-C., CHENG K. B., LEE J., LEE T.-Y.: Real-time physics-based 3d biped character animation using an inverted pendulum model. *Visualization and Computer Graphics, IEEE Transactions on 16*, 2 (2010), 325–337.

[TLT11] TAN J., LIU K., TURK G.: Stable proportional-derivative controllers. *Computer Graphics and Applications, IEEE*, 99 (2011), 1–1.

[TSF05] TSANG W., SINGH K., FIUME E.: Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2005), ACM, pp. 319–328.

[TSR01] TAKAHASHI C. D., SCHEIDT R. A., REINKENSMEYER D. J.: Impedance control and internal model formation when reaching in a randomly varying dynamical environment. *Journal of neurophysiology 86*, 2 (Aug. 2001), 1047–51.

[TYS91] TAGA G., YAMAGUCHI Y., SHIMIZU H.: Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics 65*, 3 (1991), 147–159.

[TZS04] TEDRAKE R., ZHANG T., SEUNG H.: Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proc. of the Int. Conf. on Intelligent Robots and Systems* (2004), pp. 2849–2854.

[VB04] VUKOBRATOVIC M., BOROVAC B.: Zero-moment point-thirty five years of its life. *International Journal of Humanoid Robotics 1*, 1 (2004), 157–173.

[vBE09] VAN BASTEN B. J. H., EGGES A.: Evaluating distance metrics for animation blending. In *Proc. of the 4th Int. Conf. on Foundations of Digital Games* (2009), ACM, pp. 199–206.

[vdBBH11] VAN DEN BOGERT A. J., BLANA D., HEINRICH D.: Implicit methods for efficient musculoskeletal simulation and optimal control. *Procedia IUTAM 2*, 2011 (Jan. 2011), 297–316.

[vdBGEZ*13] VAN DEN BOGERT A. J., GEIJTENBEEK T., EVEN-ZOHAR O., STEENBRINK F., HARDIN E. C.: A real-time system for biomechanical analysis of human movement and muscle function. *Medical & biological engineering & computing* (July 2013).

[vdH94] VAN DER HELM F. C. T.: A finite element musculoskeletal model of the shoulder mechanism. *Journal of Biomechanics 27*, 5 (1994), 551–553.

[VDJ99] VAUGHAN C., DAVIS B., JEREMY C.: *Dynamics of human gait*. Human Kinetics Publishers Champaign, 1999.

[vdP96] VAN DE PANNE M.: Parameterized gait synthesis. *IEEE Computer Graphics and Applications 16*, 2 (1996), 40–49.

[vdP00] VAN DE PANNE M.: Control for simulated human and animal motion. *Annual Reviews in Control* (2000).

[vdPF93] VAN DE PANNE M., FIUME E.: Sensor-actuator networks. In *ACM SIGGRAPH Papers* (1993), pp. 335 – 342.

[vdPKF94] VAN DE PANNE M., KIM R., FIUME E.: Virtual wind-up toys for animation. *Graphics Interface* (1994).

[vdPL95] VAN DE PANNE M., LAMOURET A.: Guided optimization for balanced locomotion. *Computer animation and simulation 95* (1995), 165–177.

[VHBO12] VICOVARO M., HOYET L., BURIGANA L., O'SULLIVAN C.: Evaluating the plausibility of edited throwing animations. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2012), Eurographics Association, pp. 175–182.

[VJ69] VUKOBRATOVIC M., JURICIC D.: Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering 16*, 1 (1969), 1–6.

[VvdH07] VEEGER H. E. J., VAN DER HELM F. C. T.: Shoulder function: the perfect compromise between mobility and stability. *Journal of biomechanics 40*, 10 (Jan. 2007), 2119–2129.

[vWvBE*10] VAN WELBERGEN H., VAN BASTEN B. J. H., EGGES A., RUTTKAY Z. M., OVERMARS M. H.: Real Time Animation of Virtual Humans: A Trade-off Between Naturalness and Control. *Computer Graphics Forum 29*, 8 (Dec. 2010), 2530–2554.

[WB85] WILHELMS J., BARSKY B.: Using dynamic analysis to animate articulated bodies such as humans and robots. *Canadian Information Processing Society Graphics Interface 1985* (1985), 97–104.

[WC06] WIEBER P., CHEVALLEREAU C.: Online adaptation of reference trajectories for the control of walking systems. *Robotics and Autonomous Systems 54*, 7 (2006), 559–566.

[WFH09] WANG J., FLEET D., HERTZMANN A.: Optimizing walking controllers. *ACM Trans. on Graphics 28*, 5 (2009), 168.

[WFH10] WANG J., FLEET D., HERTZMANN A.: Optimizing Walking Controllers for Uncertain Inputs and Environments. *ACM Trans. on Graphics 29*, 4 (2010).

[WGF08] WEINSTEIN R., GUENDELMAN E., FEDKIW R.: Impulse-based control of joints and muscles. *IEEE transactions on visualization and computer graphics 14*, 1 (2008), 37–46.

[WH00] WOOTEN W., HODGINS J.: Simulating leaping, tumbling, landing and balancing humans. In *IEEE Int. Conf. on Robotics and Automation* (2000), vol. 1, pp. 656–662.

[WHDK12] WANG J., HAMNER S., DELP S., KOLTUN V.: Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. on Graphics 31*, 4 (2012), 25.

[Wie02] WIEBER P.: On the stability of walking systems. In *Proceedings of the international workshop on humanoid and human friendly robotics* (2002), pp. 53–59.

[WJM06] WROTEK P., JENKINS O., MCGUIRE M.: Dynamo: dynamic, data-driven character control with adjustable balance. *Proc. of the 2006 ACM SIGGRAPH symp. on Videogames 1*, 212 (2006), 70.

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *ACM SIGGRAPH Computer Graphics* (Aug. 1988), ACM, pp. 159–168.

[WL97] WOODS R., LAWRENCE K.: *Modeling and simulation of dynamic systems*. Prentice Hall Upper Saddle River, New Jersey, 1997.

[WP09] WAMPLER K., POPOVIĆ Z.: Optimal gait and form for animal locomotion. *ACM Trans. on Graphics 28*, 3 (July 2009), 1.

[WP10] WU J.-C., POPOVIC Z.: Terrain-Adaptive Bipedal Locomotion Control. *ACM Trans. on Graphics 29*, 4 (2010).

[WPP13] WAMPLER K., POPOVIĆ J., POPOVIĆ Z.: Animal Locomotion Controllers From Scratch. *Computer Graphics Forum 32* (2013).

[WZ10] WU C., ZORDAN V.: Goal-Directed Stepping with Momentum Control. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2010), pp. 113–118.

[XF05] XIA Y., FENG G.: An improved neural network for convex quadratic optimization with application to real-time beamforming. *Neurocomputing 64* (2005), 359–374.

[YCBvdP08] YIN K. K., COROS S., BEAUDOIN P., VAN DE PANNE M.: Continuation methods for adapting simulated skills. *ACM Trans. on Graphics 27*, 3 (2008).

[YCP03] YIN K. K., CLINE M. B., PAI D. K.: Motion perturbation based on simple neuromotor control models. *Pacific Conf. on Computer Graphics and Applications* (2003), 445–449.

[YH09] YAMANE K., HODGINS J.: Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data. In *Intelligent Robots and Systems* (Oct. 2009), IEEE, pp. 2510–2517.

[YL10] YE Y., LIU C.: Optimal feedback control for character animation using an abstract model. *ACM Trans. on Graphics 29*, 4 (July 2010), 74.

[YLS04] YANG P., LASZLO J., SINGH K.: Layered dynamic control for interactive character swimming. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2004), pp. 39–47.

[YLvdP07] YIN K. K., LOKEN K., VAN DE PANNE M.: Simbicon: Simple biped locomotion control. *ACM Trans. on Graphics 26*, 3 (2007), 105.

[Zaj89] ZAJAC F. E.: Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering 17*, 4 (1989), 359–411.

[Zei84] ZEIDLER F.: The significance of lower limb injuries of belted drivers. *Journal of Orthopedics* (1984).

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2002), ACM Press, p. 89.

[ZMCF05] ZORDAN V., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. *ACM Trans. on Graphics 24*, 3 (2005), 697–701.

[ZMM*07] ZORDAN V., MACCHIETTO A., MEDINA J., SORIANO M., WU C. C.: Interactive dynamic response for games. In *Proc. of the 2007 ACM SIGGRAPH symp. on Video games* (2007), ACM, p. 14.

[ZSC90] ZATSIORSKY V., SELUYANOV V., CHUGUNOVA L.: Methods of determining mass-inertial characteristics of human body segments. *Contemporary problems of biomechanics* (1990), 272–291.

[ZvdP05] ZHAO P., VAN DE PANNE M.: User interfaces for interactive control of physics-based 3d characters. *Symposium on Interactive 3D graphics and games* (2005), 87–94.

SAMENVATTING

Bij virtuele werelden draait alles om geloofwaardigheid. Als je kijkt naar een film, of een game aan het spelen bent, wil je volledig op kunnen gaan in de wereld die wordt voorgeschoteld. Elk detail dat niet past of klopt kan de zorgvuldig opgebouwde illusie compleet verstoren. Beweging vormt hierbinnen een belangrijk onderdeel. Om een virtuele wereld realistisch over te laten komen is het belangrijk dat alles wat beweegt zich – net als in onze wereld – houdt aan de wetten van de fysica. Wetenschappers op het gebied van computeranimatie zijn voortdurend op zoek naar manieren waarop natuurwetten kunnen worden geïncorporeerd in virtuele werelden.

Van oudsher worden animaties met de hand vervaardigd door kundige animatoren. Dit is een tijdrovend proces en het resultaat is volledig *kinematisch*, wat wil zeggen dat alles vrij kan bewegen, zonder rekening te houden met zaken als massa of zwaartekracht. De vraag in hoeverre zulke bewegingen fysisch correct zijn ligt volledig in handen van de animator.

Als alternatief wordt er tegenwoordig vaak gebruikgemaakt van *motion capture*. Bij deze techniek worden bewegingen gebruikt van echte acteurs, door ze op te nemen met speciale sensoren. Deze bewegingen zijn gegarandeerd fysisch correct – de acteur heeft zich tenslotte moeten houden aan alle natuurwetten. Een nadeel van deze methode is dat je haar alleen kunt gebruiken voor bewegingen die ook daadwerkelijk live uitgevoerd kunnen worden. Het animeren van gevaarlijke stunts, zoals springen van een steile rots, is niet mogelijk. Ook is het niet eenvoudig om op die manier wezens te animeren die er heel anders uitzien dan de acteurs.

Animatie wordt nog een stuk ingewikkelder als er sprake is van interactie. Er zijn namelijk ontelbaar veel manieren waarop karakters en objecten met elkaar in contact kunnen komen en op elkaar kunnen reageren. Vaak kan een subtiel verschil in contact zorgen voor compleet andere beweging. Denk bijvoorbeeld aan een toren van blokken die instort: als je de blokken net iets

anders omduwt kunnen ze op een totaal andere plek terechtkomen. Het is ondoenlijk om van tevoren met alle mogelijke variaties rekening te houden. Hierdoor zie je bij games dat bewegingen tijdens interacties vaak hetzelfde zijn, en niet helemaal realistisch overkomen.

Dit proefschrift gaat over een alternatieve methode om animaties te maken: namelijk door gebruik te maken van fysica-simulatie. In plaats van het handmatig ontwerpen of opnemen van bewegingspatronen, kun je deze ook berekenen door gebruik te maken van natuurkundige formules. Alle beweging die daaruit voortkomt volgt automatisch de wetten van de fysica en komt daardoor natuurlijker over.

Het idee om computers te gebruiken voor fysica-simulatie is niet nieuw. Sterker nog, de allereerste computer, de ENIAC, was ontworpen om bewegingsbanen te berekenen voor oorlogsgeschut. Een van de allereerste computeranimaties (gemaakt in 1963) is gebaseerd op een simulatie van een satelliet die om de aarde beweegt. Tegenwoordig is fysica-simulatie een zeer gebruikelijke methode voor het animeren van water, rook, kleding en levenloze wezens (zogenaamde *rag-dolls*). Alleen voor het animeren van levende wezens wordt meestal nog steeds gekozen voor meer traditionele kinematische methoden, ondanks het feit dat er al decennialang onderzoek wordt gedaan naar de mogelijkheid om ook hiervoor gebruik te maken van fysica-simulatie.

Om deze terughoudendheid te begrijpen is het belangrijk te beseffen wat er allemaal komt kijken bij deze vorm van animatie. Fysisch gesimuleerde karakters kunnen namelijk, net als wij, alleen aangestuurd worden door middel van krachten (bij ons zijn dat spieren die samentrekken). Als je ergens naar toe wilt bewegen, dan kan dat alleen door je doelbewust en gecoördineerd af te zetten tegen de buitenwereld. Hetzelfde geldt voor je evenwicht: je moet voortdurend kleine correcties uitvoeren om te voorkomen dat je omvalt. Karakters die kinematisch worden geanimeerd hebben daar geen last van: hun positie en oriëntatie kunnen elk moment direct worden aangepast – ze hoeven nooit bang te zijn om te vallen.

Fysisch gesimuleerde karakters lijken in dit opzicht heel erg op robots. De technieken die gebruikt worden om ze aan te sturen zijn dan ook vaak afkomstig uit de robotica. De nadruk ligt daarbij echter op efficiëntie en veiligheid (robots kunnen meer schade aanrichten dan animatiefiguren), terwijl binnen computeranimatie de nadruk ligt op de visuele kant van beweging. Gesimuleerde karakters die net als robots worden aangestuurd lijken daardoor vaak een beetje houterig en stijf (ofwel: een beetje robotachtig). Zij houden zich misschien wel aan de wetten van de fysica, maar negeren de biologische beperkingen waar natuurlijke wezens mee te maken hebben. Bijvoorbeeld: spieren hebben specifieke eigenschappen die ervoor zorgen dat ze maar een beperkte hoeveelheid kracht kunnen geven, afhankelijk van hoe lang ze zijn en hoe snel ze samentrekken. Zenuwbanen zijn relatief

langzaam met het doorgeven van informatie; dit zorgt ervoor dat mensen en dieren met enige vertraging reageren op zintuiglijke waarnemingen. Robots hebben geen last van deze beperkingen en bewegen daardoor van nature anders dan mensen en dieren. Als oplossing hiervoor proberen onderzoekers steeds vaker om ook deze biologische beperkingen onderdeel uit te laten maken van de simulatie. Hiervoor maken ze gebruik van resultaten uit de *biomechanica*, een onderzoeksgebied dat zich richt op de bewegingen van levende wezens.

Er komt dus veel kijken bij deze vorm van computeranimatie: wiskunde, natuurkunde, kunstmatige intelligentie, robotica en biomechanica. Het is een grote uitdaging om al deze aspecten te combineren tot een werkend geheel. Desondanks is er de afgelopen jaren enorme vooruitgang geboekt: fysisch gesimuleerde karakters kunnen steeds meer en ze zien er steeds natuurlijker uit. Het is een spannend en veelzijdig onderzoeksgebied, dat in de toekomst steeds relevanter zal worden.

Dit proefschrift bestaat uit drie delen. Na een introductiehoofdstuk ga ik in het eerste deel in op de basisprincipes achter fysica-simulatie (hoofdstuk 2) en de basistechnieken die gebruikt worden voor het aansturen van virtuele karakters binnen een gesimuleerde omgeving (hoofdstuk 3). Het tweede deel gaat over onderzoek naar de perceptie van animatie. In hoofdstuk 4 presenteer ik een methode waarmee een willekeurige animatie beoordeeld kan worden op fysische correctheid, door gebruik te maken van spierskeletmodellen die ontwikkeld zijn binnen de biomechanica. Op die manier kan een animator eenvoudig zien of animaties wel of niet realistisch zijn. In hoofdstuk 5 werk ik een methode uit die de kwetsuur van een gesimuleerd mens kan meten. Dat kan door bepaalde fysische eigenschappen binnen de simulatie te meten, en de resultaten te vergelijken met onderzoek dat is gedaan met *crash test dummies*. Het derde en laatste deel gaat over methoden om gesimuleerde karakters aan te sturen – het belangrijkste onderdeel van dit proefschrift. In hoofdstuk 6 staat een methode waarmee gesimuleerde karakters opgenomen bewegingen zo goed mogelijk na kunnen doen, zonder dat ze daarbij omvallen. Het bijzondere van deze methode is dat zij relatief eenvoudig en efficiënt is, vergeleken met de meeste bestaande methoden. In hoofdstuk 7 beschrijven ik een methode om tweebenige wezens te laten lopen en rennen, door middel van spiermodellen. Bij deze methode wordt de optimale ligging van de spieren bepaald door middel van *trial and error* – net als binnen de evolutietheorie. De methode is hierdoor zeer flexibel en kan gebruikt worden om een breed scala aan (niet bestaande) wezens tot leven te brengen. We sluiten het proefschrift af met een algemene discussie over het animeren van karakters met behulp van fysica-simulatie. In dit hoofdstuk wordt tevens verder ingegaan op mogelijke toekomstige onderzoeksrichtingen en toepassingen.

DANKWOORD

Er is een aantal mensen dat (direct of indirect) heeft bijgedragen aan de totstandkoming van dit proefschrift. Ik wil hierbij de gelegenheid nemen om hen te bedanken.

Om te beginnen wil ik mijn begeleider Frank van der Stappen bedanken. Ik heb onze aangename gesprekken altijd zeer gewaardeerd, net als je wetenschappelijke kennis en enthousiaste feedback. Mijn promotor Remco Veltkamp wil ik bedanken voor het vertrouwen en de steun die hij mij gegeven heeft, ondanks de soms ietwat hectische omstandigheden waarbinnen hij heeft moeten opereren.

Ik wil ook graag mijn leescommissie bedanken: Michiel van de Panne, Marie-Paule Cani, Marc van Kreveld, Jaap van Dieën en Frans van der Helm. Hun feedback heeft het proefschrift naar een hoger niveau gebracht. Daarvan wil ik Michiel extra bedanken voor de prettige samenwerking tijdens ons meest recente paper, net als Frans voor de feedback die hij daar onbaatzuchtig op gegeven heeft.

Ook alle andere mensen met wie ik een wetenschappelijke samenwerking heb genoten wil ik bedanken: Nicolas Pronost (mijn tweede begeleider), Arjan Egges (mijn eerste begeleider), Ton van den Bogert, Frans Steenbrink, Diana Vasilescu en Ben van Basten. Daarnaast wil ik Jan en Ewoud bedanken voor het spelfoutvrij helpen maken van de Nederlandse samenvatting van dit proefschrift.

Ik kijk met trots en tevredenheid terug op mijn promotie en ben blij dat ik dit avontuur ben aangegaan. Ik wil een aantal mensen bedanken dat mij geholpen heeft bij het zetten van deze stap. Om te beginnen is dat Michiel Westermann voor zijn hulp bij het zoeken naar mogelijkheden om te promoveren. Daarnaast zijn dat Frans van der Helm, Bert Otten en Ton van den Bogert; ik wil hen bedanken voor de enthousiasmerende gesprekken die ik met hen gevoerd heb, nog voor de aanvang van mijn promotie. Ook mijn

oorspronkelijke promotor en begeleider, Mark Overmars en Arjan Egges, wil ik bedanken voor de mogelijkheid die ze mij hebben geboden om te promoveren binnen de Universiteit Utrecht.

Op de universiteit heb ik mij altijd erg vermaakt en op mijn gemak gevoeld, hiervoor wil ik alle medewerkers en collega's bedanken. Ook buiten de universiteit heb ik mij vaak uitstekend met hen vermaakt; het best bijgebleven zijn de lange nachten vol alcoholmisbruik met onder andere Ben, Nicolas, Sander en Ioannis.

Uiteraard wil ik ook mijn familie bedanken: Joop, Edith en Lydia. Ik heb hun steun, trots, en enthousiasme altijd gevoeld en gewaardeerd, ook al waren deze doorgaans niet gestoeld op enige relevante kennis van mijn promotieonderwerp. Hetzelfde geldt voor mijn vrienden (waaronder mijn paranimfen Daan en Wouter). Ik prijs mijzelf gelukkig jullie te kennen en heb genoten van de momenten waarop jullie mijn filmpjes met omvallende computerfiguren vol verbazing en vrees hebben aanschouwd. Het gaat te ver om alle namen te noemen – gelukkig is geen van hen die ik bedoel snel beledigd.

Tot slot wil ik mijn lieve vriendin Ellen bedanken, wier schier onvoorwaardelijke steun en liefde de laatste jaren van mijn promotie vele malen mooier en makkelijker hebben gemaakt.

PUBLICATIONS

The contents of this thesis has been based on the following publications:

Chapters 2 + 3

GEIJTENBEEK T., PRONOST N.: Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. In *Computer Graphics Forum 31, 8* (2012), The Eurographics Association, 2492–2515.

Chapter 4

GEIJTENBEEK T., VAN DEN BOGERT A., VAN BASTEN B., EGGES A.: Evaluating the Physical Realism of Character Animations using Musculoskeletal Models. In *Lecture Notes in Computer Science, vol. 6459* (2010), Springer-Verlag, 11–22.

Chapter 5

GEIJTENBEEK T., VASILESCU D., EGGES A.: Injury Assessment for Physics-Based Characters. In *Lecture Notes in Computer Science, vol. 7060* (2011), Springer-Verlag, 74–85.

Chapter 6

GEIJTENBEEK T., PRONOST N., VAN DER STAPPEN A.F.: Simple Data-Driven Control for Simulated Bipeds. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), The Eurographics Association, 211–219.

Chapter 7

GEIJTENBEEK T., VAN DE PANNE M., VAN DER STAPPEN A.F.: Flexible Muscle-Based Locomotion for Bipedal Creatures. In *ACM Transactions on Graphics 32, 6* (2013).

Curriculum Vitae

Thomas Geijtenbeek was born on October 2, 1976 in Eindhoven, The Netherlands. He obtained his gymnaseum degree at the van Maerlantlyceum in Eindhoven in 1995, after which he studied Artificial Intelligence at the University of Amsterdam. He received his MSc degree in 2003, on the topic of face recognition. This was followed by a period of full-time employment as manager software development at Motek Medical, Amsterdam. In 2009, he started a PhD at Utrecht University, for which he completed his thesis in 2013.