Interactive Character Animation using Simulated Physics

T. Geijtenbeek, N. Pronost, A. Egges and M. H. Overmars

Games and Virtual Worlds, Utrecht University, The Netherlands

Abstract

Physics simulation offers the possibility of truly responsive and realistic animation. Despite wide adoption of physics simulation for the animation of passive phenomena, such as rigid objects, fluids, cloths and rag-doll characters, commercial applications still resort to kinematics-based approaches for the animation of actively controlled characters. However, in recent years, research on interactive character animation using simulated physics has resulted in tremendous improvements in controllability, robustness, visual quality and usability. In this review, we present a structured evaluation of relevant aspects, approaches and techniques regarding interactive character animation using simulated physics, based on over two decades of research. We conclude by pointing out some open research areas and possible future directions.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

1. Introduction

Responsiveness is an important aspect of computer animation. Many applications involve densely populated virtual environments, where characters and objects continuously interact with each other and with their surroundings. Proper animation of such interaction is important for the perceived realism of these virtual environments. However, creating realistic responsive animation is challenging, because the range of possible interactions is enormous, and subtle variations in initial interaction conditions may call for substantially different responses.

Kinematics-based animation frameworks rely heavily on existing motion data (either recorded or manually crafted) when generating animation. During interactions, responsive actions are selected using a carefully designed system of events, rules and scripts; the matching responsive animations are then generated through utilization of a database of motion clips. Despite great advances both in the availability and utilization of motion data, as well as in algorithms responsible for selecting appropriate response motions, this approach suffers from one major drawback: the ability to generate realistic and non-repetitive responsive animation is always restricted by the contents of the motion database.

Physics simulation offers an approach to computer animation that is fundamentally different. Instead of directly controlling the motion of virtual entities, this approach uses a *physics simulator* as an integral part of the animation loop. All motion in the virtual environment is the direct result of physics simulation, and control within the environment occurs only through the application of forces and torques – similar to real world motion. The result is that all responses of interacting entities are physically realistic by definition. In addition, subtle variations in initial interaction conditions automatically result in unique and original animations.

The possibilities of physics simulation for character animation have been recognized early on [AG85, WB85]. For the simulation of *passive* phenomena, such as rigid objects, fluids, cloths and rag-doll characters, physics simulation has been subject to wide commercial adoption, both in video games and production movies. However, despite more than two decades of research on physics-based character animation, commercial frameworks still resort to kinematics-based approaches when it comes to animating *active* virtual characters [PP10]. We can point out a number of reasons for this, which we will review below.

First of all, there is the issue of *controllability*. In physics-based character animation, the pose of a character cannot be controlled directly, but only through the application of forces and torques. Control of global position and orientation must occur through manipulation of external contacts (characters

are *underactuated*). If physics-based characters fall, getting them back up can be an enormous challenge. This also affects direct *user control*. Compared to kinematics-based approaches, control of physics-based characters is sluggish, and different from what high-paced action gamers are used to. Physics-based characters typically exhibit a high-level of autonomy, which is undesirable in such applications.

Another consideration is control of *style* and *naturalness*. Many applications require control of style, to reflect mood, personality or intention of a character. Motion capture has been an effective tool to generate a variety of natural stylistic motions. Without unexpected perturbations, kinematic animation techniques based on motion capture data generally produce higher quality animations. Especially early examples of physics-based animations qualified as robotic and stiff in comparison [NF02]. However, this has improved significantly, especially for physics-based methods that can closely track captured motion. But the main advantage of physics-based methods over data-driven methods is their ability to interact naturally to unanticipated events. In addition, motion capture animation is limited to creatures that are willing to participate in motion capture recording – humans, mostly. Dangerous stunts such as jumping down headfirst from a staircase are not suitable for motion capture, but can easily be performed by a physics-based virtual stuntman [Fal01].

A final important consideration is *usability*. In general, physics-based character control is significantly more difficult to implement and put to use than kinematics-based alternatives – even with great advances in physics simulation. To implement physics-based controllers requires knowledge of dynamics, numerical integration, biomechanics, and optimization theory. Many physics-based approaches require skillful tuning before a desired result is reached, while others require expensive off-line optimization. Such work is often *inflexible* to changes in character morphology. A final practical issue of physics-based character animation is its relatively high *computational requirements*. It may only be for about a decade that physics-based characters can be simulated on consumer-grade PCs in real-time. Still, the amount of simulated controlled characters is limited, depending on the complexity of the character and control technique.

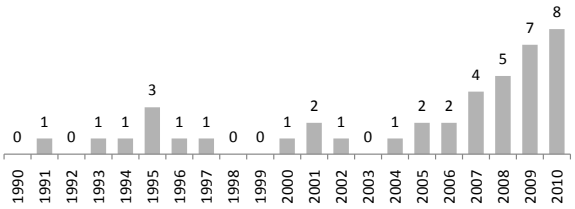

Figure 1: *The yearly number of* SIGGRAPH *and* EUROGRAPHICS *publications on interactive character animation using simulated physics.*

Recent years have shown tremendous improvements in interactive physics-based character animation. After years of focus on data-driven animation techniques, there is an apparent renewed interest in the topic of character animation through simulated physics (see Figure 1). Based on this recent trend, we expect that physics simulation will play an increasingly important role in interactive character animation in the upcoming years.

This review aims to provide a structured evaluation of different aspects, approaches and techniques regarding interactive character animation using simulated physics. It is intended both as a thorough introduction for people with an interest in physics-based character animation, as well as a reference for researchers already familiar with the subject. The remainder of this document is organized as follows. We first describe the following fundamental components: *physics simulation*, *character modeling* and *motion control*. We follow by describing different approaches in *motion controller design*. We conclude by providing an summary of different approaches and techniques, and point out possible directions for future research.

2. Fundamentals

Interactive character animation using simulated physics consists of the following three fundamental components:

1. A *physics simulator*, which is the heart of any physics-based animation system and is responsible for generating the animation, by enforcing physical laws of motion.
2. A *physics-based characters*, which are the actors in the physics simulator. They contain several physics properties usually not seen in kinematics based frameworks.
3. A *motion controller*, which can be regarded as the brain of a physics-based character: it attempts to compute the forces and torques required to perform high-level tasks.

The remainder of this section describes each of these fundamental components.

2.1. Physics Simulation

All motion in interactive physics-based animation is the result of on-line physics simulation. A *physics simulator* iteratively updates the state of a virtual environment, based on its current state, and external forces and torques (see Figure 2). Control is admissible only through application of forces and torques; the final animation is the direct result of these iterative updates.

There are many kinds of physics simulation, depending on type of physics and the level of detail that is required for a specific application. In general there is a trade-off between accuracy and performance. In interactive character animation, the focus is on efficient simulation of articulated structures. The type of simulation considered most suitable for this purpose is often referred to as *constrained rigid*

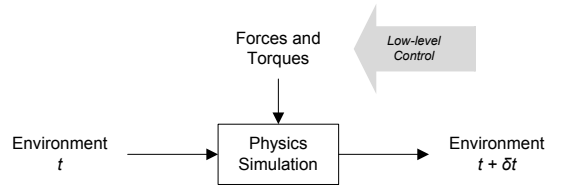

Figure 2: *Animation using physics simulation.*

body simulation, where *rigid* indicates that bodies are non-penetrable and non-elastic, and their motion is *constrained* because objects (body parts) are linked together. A physics simulator of this type performs the following operations (see also Figure 3):

1. *Collision detection* investigates if intersections exist between the different object geometries, and computes information on how to prevent further intersection.
2. *Forward dynamics* computes the linear and angular acceleration of each simulated object, considering external forces and torques, and constraints.
3. *Numerical integration* updates positions, rotations and velocities of objects, based on the accelerations found by forward dynamics.

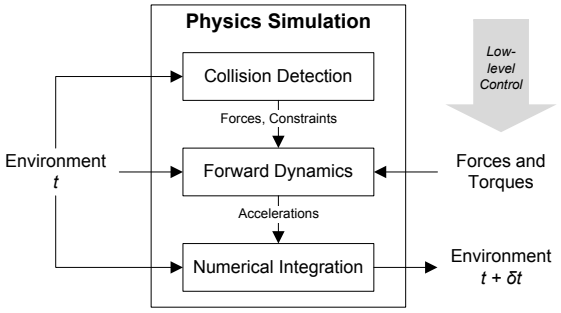

Figure 3: *Animation using physics simulation (detailed).*

2.1.1. Collision Detection and Response

There are many textbooks available that describe methods for collision detection [Cou01], all of which are beyond the scope of this review. In this section we will focus on what happens *after* a collision detected.

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

$$||f_{xz}|| \leq \mu ||f_y|| \tag{1}$$

where f_y is the normal component of the collision force, f_{xz} is the perpendicular 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.

When collision response is modeled through constraints, a symbolic link is constructed at the point of collision that restricts movement of the colliding objects. This link is removed when objects are pulled away from each other, or transformed into a sliding constraint when Equation (1) no longer holds.

The way in which friction is modeled can have a significant impact on subsequent motion control strategies, and the final motion of an interactive character [MdLH10].

2.1.2. Forward Dynamics

The goal of forward dynamics is to compute linear and angular accelerations of simulated objects, based on external forces and constraints. We will briefly describe some of the basic principles of forward dynamics, and refer to a textbook such as [Fea08] for details.

During simulation, the state of a rigid body is described not only by *position* and *orientation*, but also by linear and angular *velocity*. Change in linear and angular velocity depends on the *total mass* of an object, the location of the *center of mass*, and how the mass is distributed with respect to center of mass. The latter is often represented by an *inertia tensor* matrix. Without the application of forces or torques, *linear and angular momentum* remain constant. The relations between velocity and momentum are as follows:

$$L = mv \tag{2}$$

$$H = I\omega \tag{3}$$

where L is linear momentum, m is mass, v is linear velocity, H is angular momentum, I is the inertia tensor and ω is the angular velocity of an object. Change in linear momentum is equal to the applied force, while change in angular momentum is equal to the applied torque:

$$\dot{L} = F \tag{4}$$

$$\dot{H} = \tau \tag{5}$$

where F is a force and τ is a torque. These forces and torques can be the result of external contact, gravity, inter-object constraints or *actuation* (e.g. muscles).

Inter-object constraints are represented by *joints*. Example joint types are the *hinge* joint, which only rotates around one axis (often used to model knee or elbow joint) and the ball-and-socket joint, which rotates around three axes (often

used to model shoulder or hip joint). In dynamics simulation, these joints restrict the motion of the bodies through the application of *constraint forces*. The dynamics of a connected set of bodies can be formulated as:

$$M(q)\ddot{q} + c(q,\dot{q}) + T(q)\tau + e(q) = 0 \quad (6)$$

where q is the vector of generalized degrees-of-freedom (DOFs) of the system, $\dot{q}$ and $\ddot{q}$ are velocity and acceleration of these generalized DOFs. $M(q)$ is a pose-dependent matrix describing mass distribution. τ is the vector of moments and forces acting on the generalized DOFs. The vector $c(q,\dot{q})$ represents internal centrifugal and Coriolis forces. The vector $e(q)$ represents external forces and torques, caused by gravity or external contact. The matrix $T(q)$ is a coefficient matrix, whose form depends on $M(q)$. Algorithms for constructing $M(q)$, $c(q,\dot{q})$, $T(q)$ and $e(q)$ are described by Kane and Levinson [KL96].

In forward dynamics, the goal is to find $\ddot{q}$:

$$\ddot{q} = M(q)^{-1}\left[c(q,\dot{q}) + T(q)\tau + e(q)\right] \quad (7)$$

2.1.3. Numerical Integration

After the generalized accelerations, $\ddot{q}$, are known, they must be integrated to acquire updated velocity and position. A numerical integrator g updates generalized coordinates q_t at time t with a step of δt:

$$q_{t+\delta t}, \dot{q}_{t+\delta t} = g(\delta t, q_t, \dot{q}_t, \ddot{q}_t) \quad (8)$$

For details on numerical integration, we refer to [WL97].

2.1.4. Inverse 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*:

$$\tau = T(q)^{-1}\left[M(q)\ddot{q} + c(q,\dot{q}) + e(q)\right] \quad (9)$$

This process is used frequently in biomechanics to analyze the motion of humans, with motion data that is augmented with external force and moment measurements [RDS*03, DAA*07, EMHvdB07, vdBGEZ07]. However, inverse dynamics can also be useful in motion control, to find the torques required to achieve a desired acceleration. Such accelerations can be derived directly from kinematic data [YCP03], or they can be the output of a motion control system [HMPH05, MZS09].

It is also possible to combine forward and inverse dynamics. For instance, it is possible to control one half of a virtual character kinematically, use inverse dynamics to compute torques required to perform that motion, and perform forward dynamics to compute the motion of the other half. Such a process is sometimes referred to as *mixed dynamics* [Ott03, vWvBE*09].

2.1.5. Available Physics Simulators

There are several readily available software libraries that implement collision detection, forward dynamics, and numerical integration in a single package. Some of these are:

- *Open Dynamics Engine* [Smi06] (ODE). An open-source dynamics engine, used in commercial games and the most popular physics simulator in research. It is continuously maintained and considered a stable.
- *PhysX* (www.nvidia.com). Formely known as Novodex by Ageia and recently adopted by nVidia, this engine is the most widely used engine in commercial games. Recently, it has also been used in research. It is currently free for non-commercial use.
- *OpenHRP* [KHK04]. A physics engine developed with a focus on robot simulation, used in research on humanoid robotics.
- *Havok* (www.havok.com). Another popular engine in entertainment, which has so far not been used in research. It is free for non-commercial use.
- *Bullet*. Another open-source physics simulator.

An overview of the performance of several of these engines has been performed by Boeing and Bräunl [BB07].

Next to these physics simulators, there are also packages that aid in setting up the equations of motion, i.e. Equation (6), for a specific character model. Examples of such packages are *SD / Fast* [HRS94] and *Autolev* [KL00]. Once constructed, such dynamical model allows for fast computation of multi-body dynamics. The downside is that it must be generated separately for each character, and collision detection and response are not integrated. Finally, there is also *Endorphin* by *NaturalMotion* (www.naturalmotion.com), which is a software package focused on physics-based character animation.

2.2. Physics-based Characters

Characters that act in a physics simulation must incorporate a number of attributes that are not required for kinematics-based characters. Not only do they require mass and inertia properties, they need *actuators*. In this section we describe basic principles of physics-based modeling, as well as different actuation models.

2.2.1. Character Body Modeling

Most characters used in physics-based animation are modeled after humans. Next to that, some studies model characters after animals [RH91, GT95, WP09] or robots [Hod91, RH91], and some characters are not modeled after any creature existing in nature [Sim94].

Physics-based characters are typically modeled as a hierarchy of rigid bodies, connected to various types of joints. Mass, center of mass, and inertia tensor are mostly computed using polygonal data with uniform density, based on

cadaveric data [ZSC90, De 96, VDO99]. Joints types are selected to match natural constraints opposed 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]. When modeling characters based on nature, the number of DOFs is generally reduced drastically, to increase simulation performance and help simplify the control strategy. Sometimes humans are reduced to simple biped characters, using a single body to model head, arms and trunk [YLvdP07].

Next to modeling a character as an articulated structure, it may be useful to model additional properties. An example is the modeling of passive mechanisms for energy storage and release. Raibert and Hodgins [RH91] model the padding material some creature have under their feet using non-linear springs. Liu et al. [LHP05] model the springy tissue of a character's shoe sole. Kwon and Hodgins [KH10] add sliding joints below the knee and the hip to emulate shock absorption due to soft tissues, joint compliance and ligaments.

Even though computer animation is generally 3D, researchers sometimes model physics-based characters in 2D. The advantages of using 2D is that physics simulation is faster and more stable, and that control is simpler in 2D. Researchers sometimes start development of a control strategy in 2D, and extend them later to 3D [YLvdP07].

Smart construction of a physics-based model can greatly simplify motion control. An example of this from robotics are 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]. Another example is the work of Wang et al. [WFH09], who include toe segments in their model to achieve better locomotion performance.

2.2.2. Character Actuation Modeling

To enable active control, characters need *actuators* that apply forces and torques to a character. In order for such forces to be realistic, they must originate from within the character. There are several ways to model such actuation, which we will describe below.

Muscle-Based Actuation In biological systems, actuation occurs through *muscles*, which are attached to bones through *tendons*. When activated, muscles contract, generating torques in the joints over which they operate. The amount of torque a muscle can exert is limited by its moment arm, which is pose dependent, and by its maximum force [GvdBvBE10]. The moment arm of a muscle depends on the body pose, while the maximum force depends on its relative length and contraction speed (shorter and quickly contracting muscles are less powerful). In addition to that, muscles (and tendons, in a lesser degree), have the ability to stretch, which makes them behave like unilateral springs. This is an important mechanism for energy storage and release [LHP05], which is for example used to make human jumping more efficient [AP93].

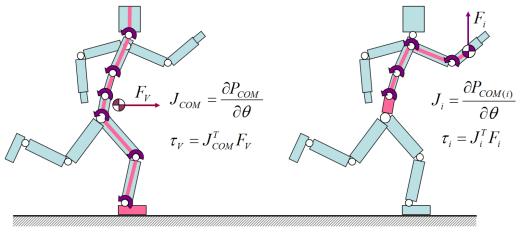

Figure 4: *Using a Jacobian transpose to transform a virtual forces into joint torques. From [CBvdP10]*

The number of muscles required for controlling an articulated structure is generally higher than the number of DOFs. Since muscles can only pull, at least two muscles are required for a single DOF. Such muscles are called *antagonistic muscles*.

In biomechanics, muscle-based actuation models are quite common [LS86]. In physics-based character animation, use of muscle-based actuation models is limited, but existing [GT95, SKP08]. Even with todays hardware, complex muscle models can be expensive computationally, real-time interactive performance is often not possible [WGF08].

Servo-Based Actuation The most commonly used actuation model assumes there is a servo motor in each joint, directly controlling each actuated DOF. In this model, the character is regarded as a robot. The advantage of such actuation model is that control is straightforward and intuitive. A downside of simplified actuation models like this is that they can lead to unnatural behavior when used in optimization methods [LHP05].

In servo-based actuation models, maximum torques are often estimated per DOF, and set to fixed values [LWZB90, KB96, OM01]. Often, however, maximum torques are not explicitly enforced; it is then up to the motion controller to ensure torques are within proper limits.

Virtual Forces Another method to actuate characters is through the application of *virtual forces*. This method is closely related to a control strategy called *virtual model control* [PCTD01], and it computes the joint torques that imitate the effect of applying a virtual force at some point on the body.

Virtual forces are transformed to joint torques using a *Ja-*

cobian transpose. In the most generic sense, a Jacobian matrix describes the linear relationship between the derivatives of two properties; in motion control, Jacobians describe the relation between change in position of a point on the character, and change in orientation of a set of joints:

$$J = \frac{\partial P}{\partial \theta} \tag{10}$$

where J is the Jacobian matrix, P a position and θ the set of joint orientations. A force F applied at position P can be translated back into a torque vector τ using:

$$\tau = J^T F \tag{11}$$

This process is further illustrated in Figure 4. Note that the range of possible virtual forces is limited because of underactuation.

External Forces Previous examples assume that characters are actuated through internal forces or torques, similar to real-world characters. In computer animation, this restriction is not strictly necessary; it is also possible to apply forces or torques on unactuated DOFs, such as global translation and rotation. Such a force is sometimes referred to as the *hand-of-God*, to emphasize the supernatural aspect of such control method [vdPL95]. Using such methods, control can be greatly simplified, at the price of a loss of naturalness. Early examples of physics-based character animation often use such external forces to control characters [Isa87, BB88]. In later research, external forces have been used to complement internal torques [WJM06].

2.3. Motion Control

In most cases, direct control of a character's actuated DOFs is highly impractical – especially for high-dimensional humanoid characters. To enable intuitive high-level control, these low-level actuation parameters are abstracted away using a so-called *motion controller*. The main task of a controller is to produce *actuator data* using environment-based *sensor data* (see Figure 5). The actuator data, which usually consists of joint torques, is then send to the physics simulator. To enable interactive high-level control, motion controllers expose a set of high-level control parameters. Examples of such parameters are speed, heading, target location or motion style.

2.3.1. Sensor Data

Motion controllers use sensor data to adapt the current behavior based on measurements. In control theory, such form of control is called *closed-loop control* (as opposed to *open-loop control*). In character animation, open-loop control is rarely used. Even the most simple control strategies (such as [vdPKF94]) use some form of feedback. In the remainder of this section, we provide an overview of often used sensor data.

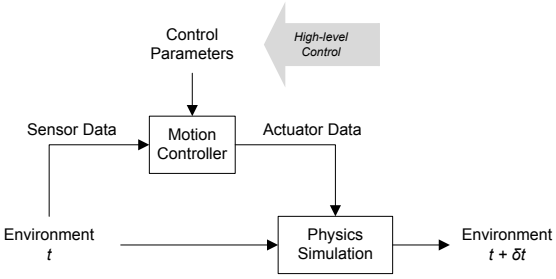

Figure 5: *The incorporation of a motion controller.*

Joint State Most motion controllers require joint orientation and velocity as feedback parameters.

Global Orientation The facing direction and up vector of the pelvis, indicating heading and leaning of a character.

Contact Information Balance and locomotion control can greatly benefit from contact information. Such information can come in the form of a boolean *on* or *off* state, or in the form of a *support polygon*, which describes the geometry of a character's base-of-support.

Center of Mass (COM) The position of the center of mass is an important quality for maintaining balance. For instance, several control strategies aim to maintain balance by keeping the projected COM inside the support polygon.

Center of Pressure (COP) The center of pressure is regarded as the point of origin of the combined ground reaction force. This point is always located inside the base of support.

Angular Momentum Another important high-level physical quantity is the angular momentum of the entire body. Several control strategies attempt to maintain balance by minimizing the angular momentum [GK04].

Zero-Moment Point (ZMP) The zero-moment point is defined as the point from which a ground reaction force would result in a zero net moment. A character is statically balanced if its ZMP coincides with its COP [Wie02].

Target Position In interactive control tasks, it is useful to know the relative position of an interactive target.

2.3.2. Motion Controller Design

Motion controller design is a multidisciplinary research topic, with ties to biomechanics, robotics, artificial intelligence and optimization theory. It can be explored from many different angles, and researchers show no consensus on how to organize the various approaches that have been developed so far. After careful deliberation, we have decided

to make a distinction between *joint-space motion control*, *stimulus-response network control* and *optimization-based motion control*. In addition to these basic approaches to motion control, we will also discuss the topic of *meta control*.

Joint-Space Motion Control (Section 3) This approach attempts to control characters in local joint-space, by defining kinematic targets for each actuated joint, and by using low-level motion controllers to compute the torque required to achieve such target.

Stimulus-Response Network Control (Section 4) This approach attempts to achieve motion control by using a generic control network to construct explicit relations between input sensors and actuators. Such control frameworks typically contain no a-priori knowledge of the control task or character, and rely heavily on optimization. They perform a lengthy off-line optimization procedure to find the set of control framework parameters for which the controller performs optimally, according to a high-level *fitness function*.

Optimization-Based Motion Control (Section 5) This method aims to construct a dynamics model of the character and its environment, and computes a set of joint torques by solving dynamics constraints with regard to a set of high-level objectives. The main difference with previous approaches is that those perform *off-line* optimization to find a set optimal *control parameters*, whereas this approach performs *on-line* optimization to find the set of optimal actuator values.

Meta Control (Section 6) In addition to the control frameworks mentioned above, there is also the topic of combining existing motion controllers to achieve concatenated, higher-level behaviors. It also describes the idea of alternating physics-based and kinematics-based control, to get the best of both worlds.

2.3.3. Control Tasks

There are several tasks a motion controller can perform, including balance, locomotion, and environment interaction tasks. However, the main focus of research in physics-based character control is on *biped locomotion* tasks, such as walking and running. In this section we will describe some aspects of locomotion.

Locomotion Cycle To simplify biped locomotion control, it is often helpful to identify the different phases and events that occur during locomotion. Common events in human locomotion are *heel-strike* and *toe-off*, to indicate the beginning and end of foot-contact. During walking, distinction is made between *single-stance* and *double-stance*, to indicate the number of feet that are in contact with the ground. For running behaviors, phases are often divided into *flight* and *contact* phase.

Inverted Pendulum Model This model is often used in biped control, to model the behavior of the center of mass during single-stance phase. It is used to predict the COM trajectory, and to compute the target position of the swing leg [TLC*09, CBvdP10]. An extension to this model is the *Spring-Loaded Inverted Pendulum* (SLIP), which models the stance leg using a spring, allowing for length variation that is seen in biped locomotion [MdLH10].

2.3.4. Evaluation

In our review, we will conclude the description of each approach with an evaluation. This section describes the criteria used in these evaluations.

Skills Repertoire The first evaluation criterion is to look at the skillset that has been developed using a given motion control approach. In our review, we focus on *balance*, *locomotion* and *environment interaction*. Balance and locomotion can be regarded as the basic skills of any physics-based interactive character. Balance is an essential sub-skill of any control task, unless this task explicitly requires loss of balance (e.g. falling). Locomotion includes any method of getting from one place to another, including walking, running, swimming, flying, etc. However, the focus in this paper will be on biped running and walking behaviors.

Many applications employ virtual worlds that contain environmental constraints, and are filled with interactive items. Skills that deal with such constraints include stepping over ridges, traveling narrow pathways, etc. Examples of interactive skills are picking up or pushing around objects, or interacting with other characters.

Robustness For many applications it is important that motion controllers are robust to unanticipated variations in the environment. Examples are unexpected perturbations, unevenness in terrain, or changes in friction coefficient.

Style and Naturalness Many applications require control of style, to reflect mood, personality or intention of a character. A common way to control style is to incorporate kinematic motion data as a reference. Next to that, it is important motions appear *natural*. Such qualifications are difficult because of their subjective nature, and the number of user studies performed to quantify the visual quality of character animations is limited.

In physics-based animation, naturalness is controlled using three basic approaches. First, by using reference data that is considered natural. Second, by incorporating motion attributes that are considered natural, such as gait symmetry [BSH99], passive knee usage during gait [Nov98, WFH09] or passive arm swing [dLMH10]. Finally, through optimization of high-level goals, such as energy efficiency. Such optimization methods, however, do not guarantee natural behavior, with the exception of high-energy motions such as diving

or jumping [LHP05]. This has been attributed to the fact that most simplified models in computer graphics have different optimums than biological models, because they ignore elasticity of muscles, tendons and ligaments [LHP05], as well as the preference to use some degrees of freedom over others [FKS*02]. In addition to that, characters may have other objectives than energy minimization. For instance, a happy character walking behaves differently from a sad one, and both probably consume more energy than needed.

User Control Playable game characters often require direct and responsive control. For physics-based characters, such level of control is challenging, since high-level control tasks must be translated into low-level actuation data. Nevertheless, several research has been performed with specifically this idea in mind. Example interaction tasks include control of speed and heading of a character, as well as setting a target location for a character to move to.

Usability As a final characteristic, we will evaluate how easy it is to employ a motion control framework in a real-world application. We will evaluate usability through *ease-of-implementation*, *flexibility* and *computational requirements*. Many motion control strategies achieve impressive results but require enormous skill and effort to implement. Other techniques may require preprocessing or tuning before they function properly. These efforts are often inflexible to changes in character morphology, which means a lot of work is required each time a new character needs to be controlled.

3. Joint-Space Motion Control

Joint-space motion control attempts to control physics-based characters by defining *kinematic target trajectories*, and by using *local feedback control* to minimize the difference between the current and desired state. It has its origin in robot control and is the most common control method in industrial robotics [KSnL05].

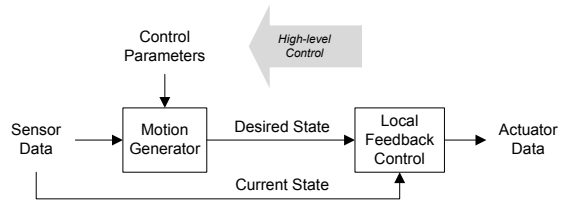

Figure 6: *Joint-space motion control*

Figure 6 shows a schematic overview of joint-space motion control for physics-based character animation. It consists of two main elements: a *motion generator* and a set of *local feedback controllers*. The motion generator is responsible for generating the desired kinematic state of each joint, usually in the form of a desired joint orientation, sometimes augmented with a desired rotation velocity. The local feedback controllers are straightforward components that compute a joint torque based on the measured displacement. Since these controllers contain little intelligence, the key challenge in joint-space motion control is to generate proper kinematic targets.

The main appeal of this approach is its simplicity: all motion can be specified in the kinematic domain, based on kinematic insights, without the need to explicitly deal with the complexities of multi-body dynamics. The downside of this approach is reduced control, caused by local feedback controllers not operating *as a whole*. Coordinated motion is the product of all joints working together, while local feedback controllers operate individually.

The remainder of this section is divided as follows. We first describe the basic principles behind local feedback control, as well as some feedback control techniques (mostly *proportional-derivative control*). Next, we describe different strategies for generating kinematic targets. We conclude this section with an evaluation of joint-space motion control frameworks.

3.1. Local Feedback Control

The purpose of local feedback control is to minimize the difference between the current state and desired state of an actuated joint, by computing a torque that is in some way proportional to the measured amount of displacement. Kinematic targets are usually represented through desired joint orientations, but may also include desired joint velocities. Even though most characters only contain rotational joints, feedback control strategies can be applied similarly to prismatic joints.

3.1.1. Proportional-Derivate Control

The feedback control method that is (by far) the most widely using in joint-space motion control is called *proportional-derivative control*, or *PD Control*. It computes a torque that is linearly proportional to the difference between the current state and a desired state. It takes into account both how far a joint orientation is from its target, as well as how fast it is currently moving. It can be formulated as follows:

$$\tau = k_p(\theta_d - \theta) + k_v(\dot{\theta}_d - \dot{\theta}) \qquad (12)$$

where τ is the final torque or force, θ is the current joint angle, θ_d the desired joint angle, $\dot{\theta}$ the current joint velocity, and $\dot{\theta}_d$ the desired joint velocity. The values k_p and k_v are called *controller gains*, and they control how reactive the controller is to differences in position and velocity, respectively.

Finding correct values for k_p and k_v 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, the controller will follow a target trajectory too rigidly and become unresponsive. This variation in stiffness is also called *impedance*. In skilled human motion, impedance is usually low [TSR01, Hog90].

Individual gains must also be set in a proper relation to each other. A relatively high value for k_p may result in *overshoot* (meaning that the joint will move past its desired position before reaching it), while a low value may cause unnecessary slow convergence. If the desired state and dynamics characteristics are fixed, the optimal relation between the two gain parameters is $k_v = 2\sqrt{k_p}$ (such controller is said to be *critically damped*). However, in character animation the desired position and velocity change invariably over time, as do the dynamics characteristics. The critical damping relation is still often used as a starting point though. Zordan and Hodgins [ZH02] scale controller gains in accordance to an estimate of the moment of inertia of the chain controlled by a joint. This effectively decreases the amount of tuning required. More details on PD gain tuning can be found in [ACL*05]. Gain parameters can also be tuned automatically through empirical optimization [vdP96].

Many applications only specify a target joint angle and use zero target velocity ($\dot{\theta}_d = 0$). Such a system is similar to a spring-damper system, where a spring is generating a force to move to its rest position. In that case, θ_d is the rest position or set point of the spring, k_p defines the spring gain and k_v the damping.

PD control does not explicitly take into account gravitational force. This may result in an error between actual and desired position, even at steady state [NF02]. In robotics, such error is often compensated by adding an integral component to the equation, which compensates for accumulated error between current and target position (so-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 integral error.

3.1.2. Antagonist Control

Antagonist feedback control is inspired by muscle-based actuation [NF02]. For each DOF, a pair of antagonistic springs operate in opposing direction to create joint torques, and also to regulate impedance [Hog90]. Each spring has a fixed set point, located at either joint limit. The control strategy can be formulated as:

$$\tau = k_L(\theta_L - \theta) + k_R(\theta_R - \theta) + k_v\dot{\theta} \qquad (13)$$

where τ is the target torque, θ is the target angle, θ_L and θ_R are the spring set points, k_L and k_R are the spring gains, and $k_v\dot{\theta}$ is the damping term.

Instead of setting a target angle, this method varies the spring gains to achieve a desired position. The relation between the spring gains can be calculated based on a target angle θ_d and a known external force F:

$$k_L(\theta_L - \theta_d) + k_R(\theta_R - \theta_d) + F = 0 \qquad (14)$$

This relation can be seen as a line, tension is controlled by selecting a point on this line. Since both springs have linear control, there exists a mathematical equivalence between PD control and antagonist control. The advantage of antagonist control is that its parameters are more natural [NF02].

3.1.3. Alternatives

We will finally mention two other alternatives to PD Control, without describing them in detail. The first is called *non-linear force fields* [MI97, MWDM98, MZW99], which is a non-linear feedback control method, modeled after principles from biomechanics research. The second is called *Model Reference Adaptive Control*, or MRAC [KMB96], uses a reference model to calculate the torque based on a desired convergence speed. We refer to cited papers for more details.

3.2. Motion Generation

The motion generator is the key component in joint-space character animation. It is responsible for generating target trajectories that lead to proper behavior. In this section, we will describe the basic approaches to generating such target kinematic trajectories, along with strengths and weaknesses of each approach.

3.2.1. Procedural Motion Generation

Analogous to kinematic animation, early attempts in joint-space motion control generate motion procedurally. Such procedures are hand-crafted attempts to model behaviors detected in biological systems, based on insights from robotics and biomechanics. They are often designed for specific behaviors, with specific characters in mind.

An early example of this approach is the work of Raibert and Hodgins [RH91], which is based on earlier robotics research [Rai86]. Their motion controllers serve as excellent illustrations of joint-space motion control with procedurally generated kinematic targets. The focus of their research is on constructing various gait types for a number of low-dimensional characters. Their character models include passive spring mechanisms and padding material to simulate energy storage-and-release. This enables them to perform a hopping style motion without explicit control. Their control system is governed by a finite state machine, which tracks the current phase of the gait cycle, based on foot contact information. During each phase, a collection of non-hierarchical control systems, each responsible for a specific task, is combined to create a single kinematic target pose. One control system uses a simplified inverted pendulum model in combination with inverse kinematics to determine target foot placement during swing phase. The foot

placement not only covers 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 system attempts to maintain the upper body in an upright position, by applying a torque to the hip or knee joint of the stance leg. This torque is proportional to the displacement of the upper body orientation in the world coordinate frame.

Using similar techniques, Hodgins et al. [HWBO95] have created full body human animations of a character running, cycling and vaulting. 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. This mimics a natural behavior called *ground speed matching*. They also add parameters to interactively control heading, by adjusting foot placement. In later research, Hodgins and Pollard [HP97] extend these controllers to work with scalable character length and mass.

In subsequent research, similar techniques have been employed to create a wide range of behaviors. Wooten and Hodgins [WH00] have created controllers for leaping, tumbling, landing and balancing. Their balance controller produces torques in knees and ankles, proportional to the difference between the projected center-of-mass and the center of the support polygon. Faloutsos et al. [FvdPT01, Fal01] demonstrate controllers for sitting, falling, rolling over and getting back up.

Despite these successes, this approach has a couple of drawbacks. First, designing procedural controllers is a skillful process that relies on intuition, experience, and laborious trial-and-error. Once developed, these controllers often not easily be applied to different characters or in different environments. Resulting motions also appear significantly less natural than motion captured equivalents, and style is also difficult to control.

3.2.2. Pose-Control Graphs

Pose-control graphs can be regarded as the physics-based equivalent of keyframe animation. In this approach, each state in a finite state machine is linked to an explicit target pose (see Figure 7). Such poses remain fixed over a longer period of time, and the motion is the direct result of local feedback controllers gradually working towards that target.

Using a pose-control graph framework, users can quickly alter the behavior of a controller by editing key poses. New behaviors can be generated similarly, without the need to explicitly consider a motion's underlying principles. However, unlike kinematic keyframe animation, poses defined in a pose-control graph will not necessarily be reached during simulation. In addition, to get proper transitions between controllers, the low level feedback controllers must be tuned in pair with the target poses. Finally, key poses often only contain a target position, because setting target velocities is

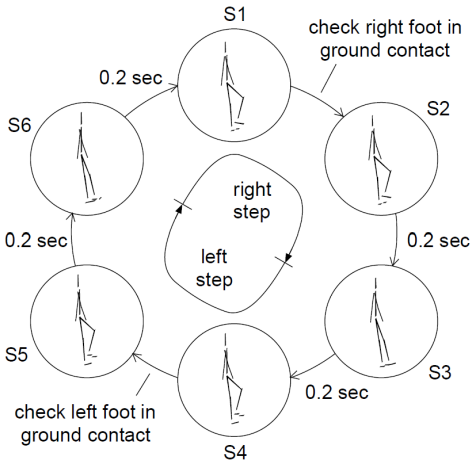

Figure 7: *Example of a pose-control graph for walking. From [LvdPF96]*

much less intuitive. To compensate for zero desired velocity, targets often need to be defined as extreme poses, far beyond an intended pose.

One of the early examples that use a pose-control graph in simulated physics is the work on *virtual wind-up toys* by Van de Panne et al. [vdPKF94]. In this work, they develop locomotion controllers for a set of low-dimensional 2D toy characters, using a state machine that advances at fixed time intervals, together with PD control. The motion generator does not incorporate any feedback from the environment or explicit balance strategy.

When controlling more complex characters (such as 3D humanoids), the amount of behaviors that can be achieved using only target poses and PD controllers is limited. There is need for additional balance control when performing tasks such as biped locomotion. Laszo et al. [LvdPF96] assume there is a linear relation between the state of a character and the control adjustments required to maintain balance. This approach (called *limit-cycle control*) has resulted in stable 3D walking controllers.

Yin et al. [YLvdP07] have developed a generic pose-control graph framework called SIMBICON (an acronym for SIMple BIped CONtroller), which allows biped locomotion control with a large variety of gaits and styles. Example controllers can walk in different directions, and perform gaits such as running, skipping and hopping. Most controllers operate with as little as four states for walking, and two states for running. The key of this control framework is an efficient balance control strategy, which consists of two elements (see Figure 8). First, it tracks target poses of both the torso and the swing leg with regard to the world frame. This provides posture control for the torso (using the stance hip torque), and makes sure the target swing foot position is independent

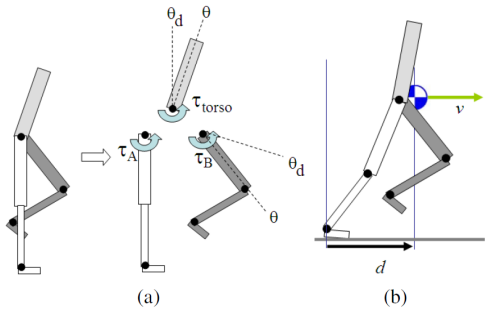

Figure 8: *The SIMBICON balance strategies. From [YLvdP07]*

from the current torso angle. Second, the desired swing hip angle is continuously adjusted to correct swing foot placement. This adjustment is proportional to the distance between the stance ankle and the projected COM, as well as the velocity of the COM.

The SIMBICON framework has been the basis for a lot of additional research. It has been used within an optimization framework to enhance environment interaction [YCBvdP08], style [WFH09], and robustness [WFH10]. It has also been extended by adding control policies on top of the framework, that have been developed off-line by systematically exploring the effect of specific controllers during various states at the beginning of a step. This has resulted in the ability to walk on constrained terrain [CBYvdP08], and a control policy that can perform tasks with long-term goals, with increased robustness [CBvdP09].

Coros et al. [CBvdP10] extend and modify the SIMBICON framework in a number of ways. First, their motion generator produces continuous target trajectories, which are specified using *Catmull-Rom* splines. Second, they use a full inverted pendulum model to achieve balance, effectively making balance control independent from body height [TLC*09]. Next, they apply *virtual forces* (as described in Section 2.2.2) to compensate for gravity, to control speed, and for performing balance control through manipulation of ground reaction forces. Finally, they incorporate the ability to manipulate objects during locomotion. A limitation of their framework is that it is only suitable for low energy locomotion behaviors, and not for running.

3.2.3. Data-driven Motion Generation

When animating humanoid characters, motion capture is an efficient way to acquire rich and natural kinematic trajectories. Since the source of the data is continuous, these trajectories include velocity information. Recorded motions are also known to be physically feasible.

However, joint-space motion control is often unable to robustly track recorded kinematic target trajectories without modifications. This is because of a number of reasons. First, a physics-based character is never exactly identical to the performing actor. Minor differences in body morphology can lead to crucial difference in motion. For instance, minor variation in leg length can cause the foot of a physics-based character to unexpectedly touch the ground while in the middle of a swing phase, causing it to trip [SKL07]. A second problem is that some of the actor's feedback mechanisms are too subtle to capture, or may only work in a specific environment. Such errors can accumulate and cause a character to become unstable. Also, motion capture data does usually not include ground reaction forces, which means not all relevant information is present. Finally, the local nature of joint-space controllers can lead to an accumulation of errors, especially in tracking of unactuated global translation and rotation. Optimization-based methods of section 5 have been more successful in robust tracking of motion capture data.

Several techniques have been developed to cope with balance issues. For non-locomotion behaviors, motion capture sources have been used in combination with a procedural balance strategy, while only tracking the upper body [ZH99]. In later work that performs full body tracking, Zordan and Hodgins [ZH02] incorporate an additional balance strategy that attempts to control the center of mass using a virtual force (see Section 2.2.2). Sok et al. [SKL07] developed a method that can 'fix' kinematic target trajectories for 2D characters, using non-linear displacement mapping. The displacement parameters are found using an off-line trial-and-error optimization process. Yin et al. [YLvdP07] have used the SIMBICON framework for animating locomotion based on motion capture data. Their approach has some limitations, because the motion data first needs to be analyzed and smoothed. Tsai et al. [TLC*09] extend this approach by using an inverted pendulum model, but their approach uses high-gain tracking.

Even though high-gain tracking in combination with continuous data leads to faithful reproduction of the original motion, it also reduces responsiveness, causing stiff behavior during unexpected perturbations. To deal with this, Zordan and Hodgins [ZH02] use standard high-gain tracking, but temporarily lower the gain values of significant body parts when a perturbation is detected. After that, they gradually increase gain back to normal. An alternative approach is to include feed-forward torques, which is explained in Section 3.3.

3.2.4. State-Action Mapping

State-Action mapping is based on the assumption that a target pose can at any time be derived from the current pose. At any point during control, the current state is used to select a pose from an array of possible target poses, which leads to a specific action that is appropriate for that state. In this way it operates as a flexible feedback mechanism, where a specific target pose can correct a state of imbalance.

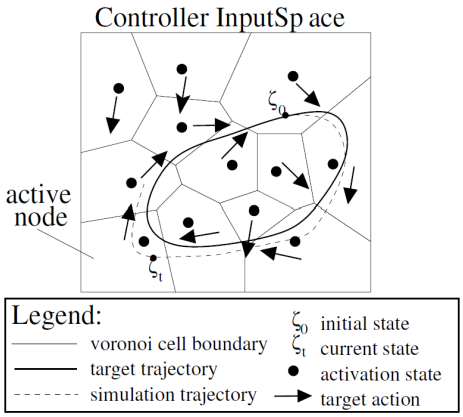

Figure 9: *State-action control. From [SvdP05]*

Mappings between current and target pose can be constructed using recorded data, where a pose at time t (the source) is mapped to a pose at time $t + \delta t$ (the target). These mappings are then further refined using an off-line empirical optimization.

An early example of state-action mapping is the work of Ngo and Marks [NM93], who describe a method called *Banked Stimulus Response* (BSR). In their work, they do not use any data to initialize their parameters, instead their approach relies solely on optimization. Auslander et al. [AFP*95] have extended their research and describe how to achieve several behaviors by defining the right set of high-level objectives.

Sharon and Van de Panne [SvdP05] have developed a similar control system that consists of a set of *control nodes*, each of which consists of an input state and a target pose (see Figure 9). At each frame, the control system selects the active node using a nearest-neighbor criterion, effectively splitting up the state space into target pose compartments. Their input state consists of joint angles, global position and global orientation, while their output state consists of joint angles. During optimization, they allow independent variation in both input state as well as in target pose.

Sok et al. [SKL07] also use a motion capture driven approach, with some key differences. First, they use a much denser set of control nodes, which are acquired by sampling several parallel cycles of motion capture data. Their input state includes velocities, foot position and ground contact information, while their target positions are augmented with target velocities. Given an input state, they select a number of nearest states, and compute a target using a weighted average of the nearest targets. During optimization, instead of directly optimizing the control node parameters, they modify the kinematic trajectories on which the control nodes are based (as described in Section 3.2.3).

3.3. Feed-forward Control

In addition to local feedback control, joint-space motion controllers sometimes include additional feed-forward torques. 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]. Yin et al. [YCP03] emulate this behavior by using inverse dynamics to compute feed-forward torques off-line, and add this to low-level PD control on-line. In later work, Yin et al. [YLvdP07] use *feedback error learning* to compute these torques. In feedback error learning, feed-forward torques are gradually derived from the feedback torques generated by PD controllers during a motion. Nunes et al. [NVCNZ08] compute feed-forward torques on-line, by using a parallel auxiliary simulation of the unperturbed motion capture data with high-gain tracking.

The main problem with these feed-forward approaches is that they do not work well with discontinuous motions. For instance, during locomotion there is a sudden increase in torque during heel strike, which is difficult to synchronize in on-line control. As a result, feed-forward methods are used only for continuous motion, or for body parts that are more or less continuous during discontinuous motion (such as the upper body during walking) [YCP03].

3.4. Evaluation

We conclude this section on joint-space motion control with an evaluation, based on the criteria described in Section 2.3.4.

3.4.1. Skills Repertoire

Early development in joint-space motion control has lead to the development of a vast set of controllers, representing several types of balance, locomotion, and interaction. There are a number of examples of basic locomotion and balancing tasks, based on procedural motion [HWBO95], pose-control graphs [YLS04, YLvdP07, CBvdP10], or motion capture data [SvdP05, SKL07].

In addition to basic locomotion and balance skills, there are several joint-space motion controllers that somehow respond to or interact with the environment. These skills have been developed either using inverse kinematics, to adjust the target trajectory of a specific end-effector, or using optimization methods. An example of the use of inverse kinematics for interaction is the work of Zordan and Hodgins [ZH02], who have developed controllers for in-place balanced characters that perform table tennis and boxing, based on motion capture data. Another example is the work of Laszlo et al. [LvdPF00], who develop controllers for climbing stairs and traversing monkey-bars. Coros et al. [CBvdP10] use a similar approach for object manipulation during locomotion.

The work of Yin et al. [YCBvdP08] is an example of environment interaction developed through optimization. They optimize the parameters of the SIMBICON framework to develop several interactive locomotion behaviors, such as stepping over objects, pushing and pulling crates, walking up and down stairs, and walking on slopes. Wang et al. [WFH09, WFH10] use a similar optimization approach to develop walking on narrow ridges. Coros et al. demonstrate constrained walking [CBYvdP08] and walking to target locations [CBvdP09], by using the optimized task control policy described previously.

3.4.2. Robustness

There have been several efforts to make joint-space controllers more robust to unexpected perturbations or variations in the environment. Most of these efforts rely on empirical optimization. An early example is the work of Auslander et al. [AFP*95], who use a randomized initial pose during optimization. In later work, the parameters of the SIMBICON controller have been optimized to achieve more robust behavior. Yin et al. [YCBvdP08] have demonstrate controllers that are robust against changes in low ground friction. In addition, Wang et al. [WFH09] optimize controllers to be robust against noise, windy environments and external perturbations. They also optimize to increase the steadiness of specific body parts (e.g. to simulate a hand holding a hot beverage). Coros et al. [CBvdP09] increase robustness by using the optimized task control policy described previously.

3.4.3. Style and Naturalness

The possibility to control style of joint-space control methods depends on the method used for motion generation. With procedural motion, style control is difficult, since the effect control parameters have on style is unintuitive. As mentioned before, procedural techniques are also not very natural. With pose-control graphs, styles can be modified more intuitively, and new styles can be authored with relative ease. The most effective example of that is of Coros et al. [CBvdP10], who allow quick control of several style elements in walking. However, pose-based methods do not offer the level of detail and naturalness of motion captured animation. Data-driven techniques demonstrate detailed style control and natural behavior, based on motion captured data. However, general purpose frameworks of data-driven using joint-space control are limited to 2D.

Both style and naturalness have been improved using optimization techniques. Wang et al. [WFH09] increase the naturalness of motion through optimization of a composite objective consisting of 8 terms, including torque minimization, head stabilization, etc. Such optimization automatically leads to different styles when character morphology changes. Also, controllers optimized for robustness automatically produce appropriate styles.

Finally, PD gain tuning is important for the perceived naturalness. Overly high gains result in motion that is both unresponsive and unnatural (e.g. [TLC*09]), while low PD gains may lead to overly loose motion. This issue can be covered by temporarily changing the gains in the event of a perturbation [ZH02], or by using feed-forward torques (see Section 3.3).

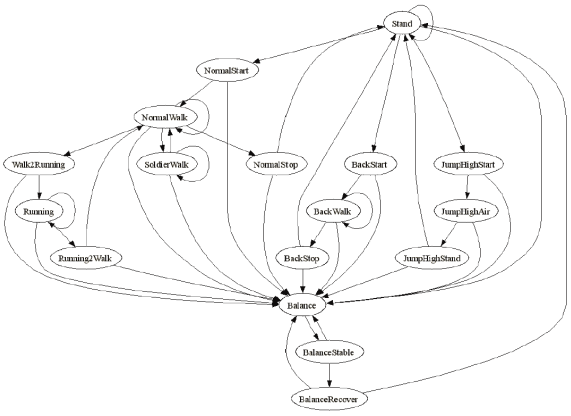

Figure 10: *Controller transitions. From [SKL07]*

3.4.4. User Control

Early procedural joint-space control methods already include some form of user interaction, such as speed control [RH91] and heading [HWBO95]. The first research focused on interactive control was that of Laszlo et al. [LvdPF00], who have developed an interactive control framework for a number of 2D characters and a great variety of behaviors. Van de Panne and Lee [VL03] develop an interactive physics-based game for simulating ski stunts. Zhao and van de Panne [ZvdP05] control diving, snowboarding and skiing in 3D using a gamepad controller. Many joint-space control frameworks allow interactive selection of behaviors [YLvdP07, SKL07], as shown in Figure 10. Shiratori and Hodgins [SH08] using accelerometer-based input devices to control physics-based characters.

3.4.5. Usability

The great advantage of joint-space motion control is its intuitiveness. The implementation of a basic joint-space control system does not require thorough knowledge of constrained dynamics, and should pose no problem for a skilled programmer. An implementation of the SIMBICON framework is readily available on-line, as is DANCE, an open-source tool for developing joint-space motion controllers [SFO05, SCAF07]. Some of the more advanced joint-space control methods, such as the *generalized biped walking control* framework [CBvdP10] are more difficult to implement, and require knowledge of inverted pendulum modeling and virtual forces.

Many joint-space frameworks are not exactly plug-and-play. They may require additional tuning before a desired behavior is achieved (such as [YLvdP07]), or extensive off-line optimization of control parameters (such as [SKL07]). Once parameters are tuned or optimized, they often work only for a specific character and behavior, unless character morphology is directly considered as input parameter [HP97]. The control framework described by Coros et al. [CBvdP10] is a good example of a more flexible framework, even though this framework is limited to walking behaviors.

Joint-space control methods generally do not have any problems performing in real-time. Both kinematic motion generation and low-level feedback control can be implemented efficiently, and take up significantly less processing time than the constrained physics simulation.

4. Stimulus-Response Network Control

This method attempts to approach the issue of motion control by emulating neural network control as found in biological systems. It assumes that, for a given control task, there exists a network-based mapping between sensor data and actuator data which can perform this task. Since manual construction of such network is impractical and unintuitive, such methods rely heavily on optimization.

Important elements in stimulus-response network control are the *fitness function*, the *optimization method*, and of course the characteristics of the stimulus-response network itself. A schematic overview of stimulus-response network control (using off-line optimization) is shown in Figure 11.

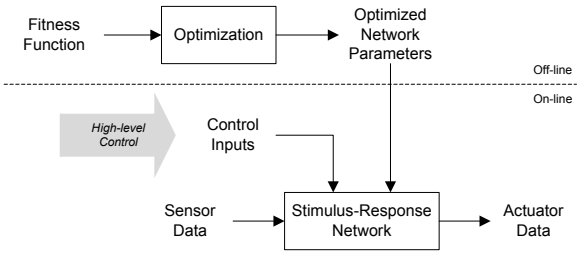

Figure 11: *Neural-network-based motion control with off-line optimization.*

This approach related to the *direct dynamics* approach described by Goh and Teo [GT88]. It is also related to the joint-space state-action mapping described in Section 3.2.4. The key difference is that in state-action control, the outputs are full body target poses, while this approach computers individual actuator values, which are related only through internal organization of the network.

The main appeal of this approach is that it allows motion controllers to be constructed automatically, without the need of any a-priori knowledge. One only needs to specify a task through a high-level fitness or reward function, and the network will organize itself to perform this task automatically. One can argue that this is similar to the way biological motion control systems develop their skills.

The challenge of this method lies in the construction of appropriate reward functions, and in the design of the control framework itself. Concerning the latter, there exists a trade-off between the flexibility of the control framework (the size of the configuration space) and the ability to optimize the framework.

4.1. Stimulus-Response Networks

Even though stimulus-response networks exist in many flavors, they have a number of features in common. First, all stimulus-response networks contain processing elements, called *nodes* or *neurons*, and links that connect these elements together. Second, each node has a single output and a number of inputs. There are *sensor nodes*, which are connected to the sensor data of a motion controller, and *actuator nodes*, which produce the actuator data.

In a traditional artificial neural network, each node outputs a value based on the weighted sum of its input nodes and a *threshold function* [Bis94]. In control, networks are typically *recurrent*, which means their internal connections can form a loop, allowing for the representation of an internal state. Finally, nodes include a time delay to allow for internal dynamics behaviors [vdPF93]. Reil and Husbands [RH02] use a small circular neural network with fixed topology and no input sensors. They incorporate feedback by converting target joint angles into torques in their output nodes, using PD controllers. Allen and Faloutsos [AF09] use a framework that allows the network topology to evolve as well, using the NEAT technique [Sta04]). However, partly due to the size of the parameter space, this approach has not resulted in stable biped locomotion control.

Other stimulus-response networks are more related to *genetic programs*, in the sense that its processing nodes can perform logical operations (*and*, *or*, etc.) or decision operations (*if*, *then*, *else*) [Gar91]. In addition, the networks of Sims [Sim94] uses several node types that promote periodic signals, such as sine or saw waves.

4.2. Optimization Strategies

There are two basic optimization strategies for neural networks: on-line and off-line. In on-line optimization, there is on-the-fly adaption of control parameters, based on feedback provided by a so-called *reward function*. This process is referred to as *reinforcement learning* [KLM96]. Off-line optimization works in a generate-and-test fashion, using a high-level *fitness function*. The fitness of a specific parameter instance (also called a *controller candidate*) is determined through simulation. The optimization process typi-

cally requires a large number of simulation trials before convergence.

While examples of on-line neural network optimization exist for control of low-dimensional robots (such as [TZS04, MAEC07, BT08]), research in computer animation focuses on off-line optimization methods. In robotics, such generate-and-test approach is often impractical (a notable exception being the work of Pollack et al. [PLF*00]).

4.2.1. Fitness Function Design

For locomotion, a common fitness criterion is to use the distance traveled in a specific time. Different styles of walking can be promoted by using weighted fitness terms, that for instance evaluate average height [vdPF93], or step length [AFP*95]. However, the process of capturing the characteristics of natural motion in high-level fitness terms is considered enigmatic and elusive [vdP00].

4.2.2. Optimization Techniques

An important aspect of optimization is the shape of the objective function, or the *fitness landscape*. The efficiency of optimization very much depends on the smoothness of such landscape. Network-based controllers tend to have an irregular landscape, with many local minima, making the optimization problem a very difficult one. To tackle this problem, researchers use a form of randomized search (sometimes referred to as *Monte Carlo* methods, or *stochastic optimization*). The random component helps the optimization to avoid quick convergence to local minima.

Most optimization techniques used for network-based controllers are based on *evolutionary algorithms* (EAs) – although there are some exceptions, such as Van de Panne et al. [vdPF93], who use an algorithm akin to simulated annealing. The choice of using an EA is often based on a design philosophy that states that behaviors that arise from such strategy are more natural [Ale01]. However, there is little evidence to support this claim, there have not been studies on network-based motion control that compare different optimization methods. It is at least questionable whether EAs are the most efficient way to optimization motion controllers, since these methods rely heavily on fitness evaluation, which is computationally expensive. Yao [Yao99] provides an overview of different approaches to using evolutionary algorithms for optimizing neural networks.

In order to increase the performance of the fitness evaluation, simulations are often cut short. This happens when a specific controller candidate is expected to perform poorly, such as a biped locomotion controller losing balance in an early stage [RH02]. Another technique is *bootstrapping*, where controllers are first optimized for simple tasks, and later for more complex tasks [Gar91, SvdP05].

Good initializations are important for optimization performance [SvdP05]. However, in contrast to state-action based methods, network-based control methods are difficult to initialize properly based on motion data. As a result, they are often initialized randomly.

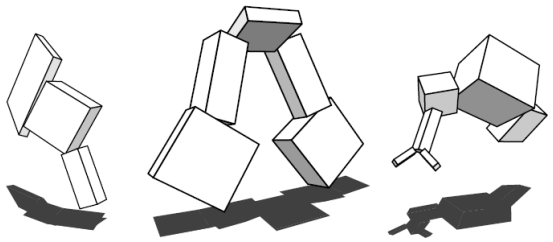

Figure 12: *Virtual creatures evolved for jumping. From [Sim94]*

4.3. Evaluation

We conclude this section with an evaluation of stimulus-response network control, based on the criteria described in Section 2.3.4.

4.3.1. Skills Repertoire

There are many examples of balanced locomotion using this approach. Early examples demonstrate locomotion behaviors for low-dimensional 2D characters [Gar91, TYS91, Tag95, vdPF93]. In addition, walking controllers have been developed for simple 3D biped characters [RM01, RH02], while running, swimming and jumping controllers have been developed for non-biped characters [Sim94, GT95]. No reports have been found on full-body humanoid biped locomotion using stimulus-response networks.

There have been several forms of interaction that have been included in stimulus-response network controllers. Examples include the following or evading of a target [Gar91, vdPF93, Sim94, RH02] and stepping over objects [Tag98]. The most interesting form of interaction is the work of Sims [Sim94], who demonstrates creatures competing over the control of an object.

4.3.2. Robustness

There are little studies that specifically deal with robustness of stimulus-response controllers. Controllers are in general considered fairly robust, unless they have been heavily optimized for specific conditions [vdPF93]. Robustness can be improved by using random variation in the test conditions during controller optimization [AFP*95]. Other studies show that including noise or random perturbation during optimization can lead to more robust controllers [WFH10], but these techniques have not been applied to stimulus-response control strategies.

4.3.3. Style and Naturalness

As mentioned before, control of style and naturalness is not straightforward when using high-level optimization goals [vdP00]. There are little studies on stimulus-response control that explicitly deal with style, or only at a very low level. We have not found any reports using this approach that use motion capture data to promote specific styles during optimization.

In spite of these difficulties, there are a number of studies that have produced striking, natural-looking animations, the most notable example being the work of Sims [Sim94]. However, these animations are limited to either low-dimensional biped characters, or character morphologies that are well-balanced by nature.

4.3.4. User Control

Research on user control is limited to controllers that are optimized to move towards an interactively controlled target [Sim94, RH02].

4.3.5. Usability

The implementation of stimulus-response network control itself is not difficult. Software libraries implementing neural networks are readily available, and custom implementations will not pose much challenge for a skilled software developer. In addition, the computational requirements of such networks (once optimized) are negligible compared to physics simulation.

However, the applications of stimulus-response network control are limited. First, it has not been successfully applied to perform full body biped locomotion tasks. Second, desired behaviors are difficult to describe in high-level goals. Finally, the optimization process can be time consuming; it may take several days before a single controller is optimized. And once optimized, such controller is inflexible to change in character morphology or environment. Any adjustment in character model, environment, or goal task requires a renewed off-line optimization procedure (although previous results can often be reused to speedup the optimization process).

5. Optimization-Based Motion Control

Optimization-based motion control frameworks attempt to compute actuator values using on-line constrained optimization. These methods formulate the dynamics of a character and its environment as a set of *constraints*, and describe the intended behavior of a character using a set of high-level *objectives*. The optimal set of actuator values is then acquired through the use of a *constraint solver*. Figure 13 shows an overview of this method.

This control method is not to be confused with optimization methods described in the previous sections. The difference is that those methods perform *off-line* optimization to find a set of optimal *control parameters*, whereas the approach described in this section performs *on-line* optimization, to find the optimal set of actuator values for a given task at a given moment.

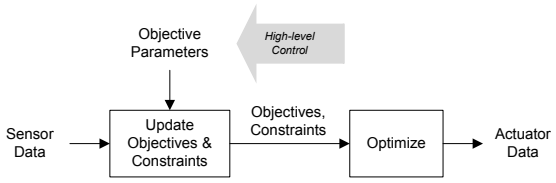

Figure 13: *Optimization-based motion control.*

The use of torque optimization owes much to the *spacetime optimization* framework as described by Witkin and Kass [WK88]. In this framework, a user defines a set of constraints (such as: a character must have a specific pose at a specific time), and a set of objectives (such as minimal energy usage), after which a physically correct motion is generated that meets these constraints, and for which the objective function is minimized. This research has been extended in many ways to generate physically realistic animations for different characters and behaviors [PW99, SHP04, ALP04, LHP05].

The spacetime approach is not directly suitable for interactive physics-based character animation using simulated physics. First, it optimizes full motion trajectories as a whole. In interactive applications this is not useful, since any unexpected disturbance invalidates the remainder of an optimized trajectory (even numerical rounding errors can count as disturbance). Second, full trajectory optimization is not feasible in real-time, even with modern hardware. Real-time interactive applications require on-line torque optimization, at each instant. The basic optimization problem can be formulated as follows:

$$\underset{\tau}{\text{argmin}} \ \{G_1, G_2, ..., G_n\} \ , \text{ subject to } \{C_1, C_2, ..., C_m\} \quad (15)$$

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, that is optimal for the given goals and constraints.

The major advantage of this approach over joint-space control is that the calculated joint torques act *as a whole*, instead of each joint trying to achieve a target by itself. This enables much tighter and more coordinated control, allowing for more robust performance of complex motions. In addition, the motion can be described using higher-level dynamics objectives, which can allow for a more intuitive problem formulation.

The downside is that, compared to most joint-space or stimulus-response methods, optimization-based motion control requires substantial skill and knowledge. One needs to be familiar with dynamics formulation and constrained op-

timization techniques. In addition, it can be challenging to design behaviors through optimization objectives, especially when objectives can interfere with each other.

In the remainder of this section, we will describe constraints, objectives and optimization strategies. We will conclude with an evaluation of optimization-based motion control.

5.1. Constraints

The most important set of constraints in optimization-based motion control enforce the Newton-Euler laws of motion during optimization. Such constraints are referred to as *dynamics constraints*, and can be directly derived from the equations of motion described in Section 2.1.2. They constrain the relation between joint torques τ and desired accelerations $\ddot{q}$. Because of their form, such constraints are called *equality constraints*.

Other dynamics constraints describe limitations on external forces and torques. Since ground reaction forces are repulsive, these limits are formulated through *inequality constraints*. An example is the Coulomb friction constraint of Equation (1). Other forms of inequality constraints enforce joint or torque limits.

5.2. Objectives

Objectives are commonly defined through *cost functions* that need to be minimized. There is a difference between objectives that can be optimized instantly at each time, and objectives that require look-ahead planning.

5.2.1. Instant Objectives

Several motion control strategies attempt to control characters through low-level objectives that can be optimized for instantly. The specification of such objectives is often state-driven. This approach has similarities with *virtual model control* [PCTD01], where articulated structures are controlled through virtual forces.

Abe et al. [AdSP07] define a low-level balance objective that drives the projected COM position towards the center of the base-of-support. Jain et al. [JYL09] define balance objectives that drive end-effectors towards support structures (such as walls or railings). Macchietto et al. [MZS09] define objectives that control angular momentum. De Lasa et al. [dLMH10] construct balancing, walking and jumping behaviors by combining several coordinated and state-driven low-level objectives. They also demonstrate walking to target positions by constructing a continuous COM trajectory objective.

5.2.2. Preview Control

Previous objectives optimize joint torques at a specific instant. This is fine when there is an incremental objective, but when tasks are more complicated, a look-ahead policy is required. Examples of such tasks are tracking of locomotion data [dSAP08a, dSAP08b, MLPP09, KH10], planned or constrained locomotion [MdLH10], or walking on uneven terrain [WP10]. When used in real-time, such planning is sometimes referred to as *preview control*.

Since it is not feasible to optimize far ahead in time using complex dynamics constraints, several methods use a low-dimensional character model for planning, and translate the results back to optimize the high-dimensional model. This approach is also used in humanoid robotics, to compute physically feasible locomotion trajectories on-line. Such trajectories are often based on ZMP control through simplified character models [KKK*03b, KKK*03a, KKI06, WC06, Che07].

In computer animation, several methods perform preview control based on motion models that have been optimized off-line. Such models provide information required for look-ahead policies during on-line optimization. Da Silva et al. [dSAP08b] use a simple three-link biped model, which is used for off-line balance optimization for a reference motion. These balance feedback torques are combined on-line with a full body tracking controller for style. Kwon and Hodgins [KH10] optimize an inverted pendulum model based on motion data to generate reference footstep coordinates on-line. Muico et al. [MLPP09] include ground reaction forces in their preview model, allowing faithful tracking of agile locomotion. Wu and Popović [WP10] model optimized end-effector trajectories and gains, to allow robust motion on uneven terrain.

Other forms of preview control use a low-dimensional character model to enable on-line look-ahead optimization. Mordatch et al. [MdLH10] use a spring-loaded inverted pendulum model (SLIP) for on-line planning of locomotion tasks on constrained and uneven terrain. Their resulting COM and foot trajectories are then tracked using low-level feature objectives described by De Lasa et al. [dLMH10].

5.3. Optimization

After all constraints and objectives are properly formulated, Equation (15) must be solved. The most common approach to tackle this is to use a quadratic program formulation, and use an on-line *quadratic solver*. We will not describe any details of this approach; instead, we refer to relevant textbooks [LY08, BHM99].

One issue that we will deal with is *objective prioritization*. Multiple objectives are often combined into a single objective function as a weighted average [JYL09]. 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]. See Figure 14 for an illustration. To get

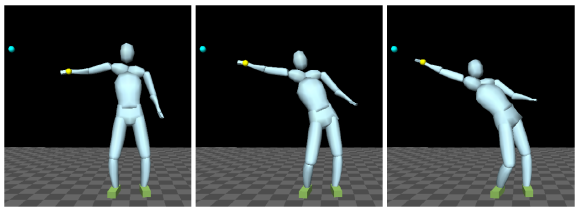

Figure 14: *Gradual increase of the weight of a reaching objective; further increment will cause the character to fall. From [AdSP07]*

around this, de Lasa et al. [dLH09] propose a solver that allows for prioritized optimization. In their method, lower-priority objectives are only minimized to a degree in which they do not interfere with higher-priority objectives.

5.4. Evaluation

We conclude this section on optimization-based motion control with an evaluation, based on the criteria described in Section 2.3.4.

5.4.1. Skills Repertoire

Many publications on optimization-based motion control focus on basic motion skills, such as balance [AdSP07, JYL09, MZS09, WZ10], walking [dSAP08a, dSAP08b, WP10], and agile running [MLPP09, KH10]. De Lasa et al. [dLMH10] have developed a control framework that, in addition to balance and locomotion, demonstrates various jumping skills. Jain et al. [JYL09] display environmental interaction in the form of object dodging, the using of walls or railings to maintain balance, or balance support for other characters. Mordatch et al. [MdLH10] demonstrate walking in a highly constrained environment, while others show walking up and down stairs and slopes [MLPP09, WP10]. Abe and Popović [AP06] demonstrate dynamic manipulation skills. Da Silva et al. [dSDP09] describe a method to combine optimization-based controllers, enabling compound skills.

5.4.2. Robustness

Optimization-based motion controllers are generally more robust than joint-space motion controllers. The main reason for this is that this approach allows actuators to work together, in a coordinated and efficient manner. This has lead to impressively robust balancing behaviors, with uneven foot placement, moving support and external perturbations [AdSP07]. In addition, recent research on locomotion control shows relatively high robustness to unevenness in terrain [MLPP09, MdLH10, WP10], and in a lesser degree to external perturbations. It is worth mentioning that many of these methods do not attempt to prevent self-collision, this is usually disabled in the physics simulator.

5.4.3. Style and Naturalness

Many optimization-based control frameworks integrate motion tracking objectives, allowing faithful reproduction of recorded motions. Especially recent publications show close and flexible tracking when compared to joint-space methods [MLPP09].

Alternatively, in methods that do not incorporate motion capture data, style can be controlled by modifying the optimization objectives. For example, De Lasa et al. [dLMH10] manually tune objectives to suggest different moods during gait.

5.4.4. User Control

Recent optimization-based motion controllers incorporate user control that is agile and direct, when compared to joint-space methods. Controllers demonstrate the ability to perform sharp turns (up to 180 degrees) with relatively short response times [MLPP09, WP10].

5.4.5. Ease-of-Use

The implementation of an optimization-based motion control framework is significantly more challenging than joint-space or stimulus-response methods. Most methods described above require thorough knowledge of constrained dynamics, contact modeling and on-line optimization techniques. Even though several existing software libraries implement these techniques, proper knowledge is still required to put these libraries to use in a motion control framework.

Once implemented, the flexibility in using optimization-based controllers with different character morphologies varies. Methods that focus on locomotion tracking require significant off-line preprocessing, and any change in motion or character morphology requires an additional preprocessing step [dSAP08a, dSAP08b, MLPP09]. On the other hand, controllers based on low-level feature-based optimization allow drastic changes to character morphology on-the-fly [dLMH10].

Finally, optimization-based methods generally have higher computational requirements than joint-space control or stimulus-response network control. The required computational power for optimization-based techniques is often on-par or above requirements for physics simulation. Computational requirements increase with the number of objectives and the degree in which they compete [JYL09], as well as the complexity of the character model, and the number of active external contacts.

6. Meta Control

The approaches described in the previous sections all focus on motion control for a single behavior. However, it may in many cases be beneficial to combine several motion controllers to get a more varied control palette, or to increase the robustness of the controlled character.

6.1. Meta-Control Frameworks

Faloutsos et al. [FvdPT01, Fal01, Fal02] have developed a meta-control framework, which automatically determines stable *pre-conditions* for controllers. Using these pre-conditions, their framework allows flexible transition from one controller to another. Their approach is independent of controller implementation, and can work with any type of controller.

Grzeszczuk and Terzopolous [GT95] use a layered approach to control. They have a set of low-level controllers that generate feed-forward activation patterns to perform specific sub-tasks, while a macro control policy combines these low-level controllers to create complex high-level tasks. Both high and low-level controllers are parameterized through optimization.

6.2. Combining Dynamic and Kinematic Animation

Many applications do not require physics-based motion control all the time. For instance, in game-like applications, it may be useful to employ kinematics-based animation during predictable phases of game-play, and switch to physics-based animation during moments of interaction. This strategy is already used with rag-doll simulation, but can also be employed with actively controlled characters.

The main difficulty is in *getting back* to kinematic control, once physical control has taken over and is finished. Zordan et al. [ZMCM05] evade this problem by generating a blending transition from the physics-based simulation to a similar recorded motion sequence, that is similar to a physics simulation, after which they are blended together, ending at the kinematic animation. Other methods include the use of external forces to control a physics-based character [YL08].

7. Summary and Future Challenges

It is an interesting time for physics-based character animation. After years of floundering, the field is rapidly maturing, and practical in-game use of physics-based character control is within reach. Undoubtedly, research in this field will continue, coming not only from animation itself but also from robotics and biomechanics. As a consequence, this review may be outdated shortly, and updated on a regular basis.

Even though there are arguments in favor of and against each of the approaches described in this review, it is clear that some approaches have a better future perspective than others. Especially the recent advance in optimization-based control methods show great promise. Joint-space methods have been around for a lot longer, but largely due to the recent impulse provided by the elegant SIMBICON framework, these methods still show improvement. In contrast, progress in stimulus-response network control is near-absent in recent years.

The main strength of joint-space control methods is its intuitiveness. New ideas for motion can easily be described in the kinematics domain, without having to directly think about the underlaying dynamics equations. In addition, joint-space control techniques are relatively easy to implement, allowing for more widespread adoption. Its main weakness is the limited coordinated control, due to the nature of local feedback control. Possible future directions include further improvement of control frameworks like that of Coros et al. [CBvdP10] to include more agile behaviors, as well as improvements in motion capture tracking techniques, which are still limited. Parameter optimization for the SIMBICON framework has led to interesting results [YCBvdP08, WFH09, WFH10], it may be interesting to apply this approach to more advanced frameworks.

The main strength of stimulus-response network control is that it allows automatic generation of controllers, based purely on high-level goals. Its main weakness is that it apparently does not scale well with the complexity of characters. Even though this control approach has resulted in striking and agile animations for low-dimensional creatures, there is no example of successful full-body humanoid locomotion control.

The main strength of optimization-based motion control is that it allows actuators to work together, and operate in an efficient and coordinated fashion. This has resulted in some of the most compelling examples of physics-based motion control to date. Since it is still relatively young, it is difficult to assess its weaknesses. One issue that may prevent widespread adoption in the research community is its complexity. Research in optimization-based control methods requires thorough understanding of physics modeling and optimization theory. There are several interesting future directions for this approach. One would be to develop a generic framework, in which several skills are combined, similar to Coros et al. [CBvdP10]. Another direction in which we are specifically interested is high-level optimization of feature-based controllers, analogous to the work of Wang et al. [WFH09, WFH10] with SIMBICON.

Another area of interest is the need for proper benchmarks that allow for quantitative comparison between controllers. Van de Panne [vdP00] has proposed the idea of *Virtual Olympics*, in which different motion controllers can compete with each other, bound by restrictions on body morphology and torque or energy usage. Another interesting idea may be to promote the entertaining aspect of controller optimization, by developing a game where the purpose is not to directly control a character, but to train autonomous physics-based characters to compete in a global event.

Finally, we see an increased tendency to use muscle-based actuation models in physics-based motion control. Such models may result in more natural motion when combined with high-level optimization objectives [LHP05], at the price of having to control several additional actuation parameters.

References

[ACL*05] ANG K., CHONG G., LI Y., LTD Y., S: PID control system analysis, design, and technology. *IEEE Trans Control Systems Tech 13*, 4 (July 2005), 559–576. 9

[AdSP07] ABE Y., DA SILVA M., POPOVIĆ J.: Multiobjective control with frictional contacts. *Proc. of the 2007 ACM SIGGRAPH/Eurographics symp. on Computer animation* (2007), 249–258. 17, 18

[AF09] ALLEN B., FALOUTSOS P.: Evolved controllers for simulated locomotion. *Motion in Games* (2009), 219–230. 14

[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 Transactions on Graphics 14*, 4 (Oct. 1995), 311–336. 12, 13, 15

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

[Ale01] ALEXANDER R. M.: Design by numbers. *Nature 412*, 6847 (Aug. 2001), 591. 15

[ALP04] ABE Y., LIU C., POPOVIĆ Z.: Momentum-based parameterization of dynamic character motion. *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), 173–182. 16

[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. 5

[AP06] ABE Y., POPOVIĆ J.: Interactive animation of dynamic manipulation. *Proc. of the 2006 ACM SIGGRAPH/Eurographics symp. on Computer animation* (2006), 195–204. 18

[BB88] BARZEL R., BARR A. H.: A modeling system based on dynamic constraints. *ACM SIGGRAPH Computer Graphics 22*, 4 (Aug. 1988), 179–188. 6

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. *Proc. of the 5th int. conf. on Comp. graphics and int. techniques in Australia and Southeast Asia - GRAPHITE '07* (2007), 281 – 288. 4

[BHM99] BERTSEKAS D., HAGER W., MANGASARIAN O.: *Nonlinear programming*. Athena Scientific Belmont, MA, 1999. 17

[Bis94] BISHOP C.: Neural networks and their applications. *Review of Scientific Instruments 65* (1994), 1803. 14

[BSH99] BODENHEIMER B., SHLEYFMAN A., HODGINS J.: The effects of noise on the perception of animated human running. In *Computer Animation and SimulationâĂŹ99, Eurographics Animation Workshop* (1999), Citeseer, pp. 53–63. 7

[BT08] BYL K., TEDRAKE R.: Approximate optimal control of the compass gait on rough terrain. *Proceedings IEEE International Conference on Robotics and Automation (ICRA)* (2008), 1258–1263. 15

[CBvdP09] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust Task-based Control Policies for Physics-based Characters. *ACM Transactions on Graphics (SIGGRAPH Asia) 28*, 5 (2009), 1–9. 11, 13

[CBvdP10] COROS S., BEAUDOIN P., VAN DE PANNE M.: Generalized biped walking control. *ACM Transactions on Graphics (SIGGRAPH) 29* (2010), 1–9. 5, 7, 11, 12, 13, 14, 19

[CBYvdP08] COROS S., BEAUDOIN P., YIN K. K., VAN DE PANNE M.: Synthesis of constrained walking skills. *ACM Transactions on Graphics (SIGGRAPH Asia) 27*, 5 (Dec. 2008), 1–9. 11, 13

[Che07] CHESTNUTT J.: Navigation planning for legged robots. *PhD Thesis, Carnegie Mellon University*, December (2007). 17

[Cou01] COUTINHO M.: *Dynamic simulations of multibody systems*. Springer Verlag, 2001. 3

[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. 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. 4

[De 96] DE LEVA P.: Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *Journal of biomechanics 29*, 9 (1996), 1223–1230. 5

[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. 18

[dLMH10] DE LASA M., MORDATCH I., HERTZMANN A.: Feature-Based Locomotion Controllers. *ACM Transactions on Graphics (SIGGRAPH) 29*, 3 (2010), 1–10. 7, 17, 18

[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. 17, 18

[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. 17, 18

[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. 18

[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. 4

[Fal01] FALOUTSOS P.: The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics 25*, 6 (Dec. 2001), 933–953. 2, 10, 19

[Fal02] FALOUTSOS P.: *Composable Controllers for Physics-Based Character Animation*. Phd thesis, University of Toronto, 2002. 19

[Fea08] FEATHERSTONE R.: *Rigid body dynamics algorithms*. Springer-Verlag New York Inc, 2008. 3

[FKS*02] FULL R. J., KUBOW T., SCHMITT J., HOLMES P., KODITSCHEK D.: Quantifying Dynamic Stability and Maneuverability in Legged Locomotion. *Integrative and Comparative Biology 42*, 1 (Feb. 2002), 149. 8

[Fro79] FROHLICH C.: Do springboard divers violate angular momentum conservation? *American Journal of Physics 47*, 0002-9505 (1979), 583–592. 5

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. *Proc. of the 28th annual conf. on Computer graphics and interactive techniques - SIGGRAPH '01*, 1 (2001), 251–260. 10, 19

[Gar91] GARIS H. D.: Genetic programming: Building artificial nervous systems with genetically programmed neural network modules. *Neural and intelligent systems integration: fifth and sixth generation integrated reasoning information systems* (1991), 207. 14, 15

[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, Citeseer, pp. 3785–3790. 6

[GT88] GOH C., TEO K.: Control parametrization: A unified approach to optimal control problems with general constraints. *Automatica 24*, 1 (Jan. 1988), 3–18. 14

[GT95] GRZESZCZUK R., TERZOPOULOS D.: Automated learning of muscle-actuated locomotion through control abstraction. *Proc. of the 22nd annual conf. on Computer graphics and interactive techniques* (1995), 63–70. 4, 5, 15, 19

[GvdBvBE10] 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 *Third International Conference in Motion in Games, Zeist, The Netherlands, November 14-16, 2010.* (Berlin, Heidelberg, 2010), Egges A., Geraerts R., Overmars M. H., (Eds.), vol. 6459 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 11–22. 5

[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 *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (2005), vol. 2, IEEE, pp. 1952–1959. 4

[Hod91] HODGINS J.: Biped gait transitions. In *Proceedings of the IEEE International Conference on Robotics and Automation* (1991), pp. 2092–2097. 4

[Hog90] HOGAN N.: Mechanical Impedance of Single-and Multi-Articular Systems. *Multiple muscle systems: biomechanics and movement organization* (1990), 149. 9

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated behaviors for new characters. *Proc. of the 24th annual conf. on Computer graphics and interactive techniques - SIGGRAPH '97* (1997), 153–162. 10, 14

[HRS94] HOLLARS M., ROSENTHAL D., SHERMAN M.: SD/Fast user's manual. *Symbolic Dynamics Inc* (1994). 4

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95* (1995), 71–78. 10, 12, 13

[Isa87] ISAACS P.: Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics, ACM SIGGRAPH Proceedings 21*, 4 (1987), 215–224. 6

[JYL09] JAIN S., YE Y., LIU C. K.: Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (SIGGRAPH) 28*, 1 (2009), 1–12. 17, 18

[KB96] KO H., BADLER N.: Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications* (1996), 50–59. 5

[KH10] KWON T., HODGINS J.: Control Systems for Human Running using an Inverted Pendulum Model and a Reference Motion Capture Sequence. *Eurographics / ACM SIGGRAPH Symposium on Computer Animation* (2010). 5, 17, 18

[KHK04] KANEHIRO F., HIRUKAWA H., KAJITA S.: OpenHRP: Open Architecture Humanoid Robotics Platform. *The International Journal of Robotics Research 23*, 2 (Feb. 2004), 155–165. 4

[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). 17

[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 *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on* (2003), vol. 2, IEEE, pp. 1620–1626. 17

[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. 17

[KL96] KANE T., LEVINSON D.: Dynamics online: theory and implementation with AUTOLEV. *Online Dynamics, Inc* (1996). 4

[KL00] KANE T., LEVINSON D.: *Dynamics online: theory and implementation with AUTOLEV*. OnLine Dynamics, 2000. 4

[KLM96] KAELBLING L., LITTMAN M., MOORE A.: Reinforcement learning: A survey. *Journal of Artificial Intelligence Research 4* (1996), 237–285. 14

[KMB96] KOKKEVIS E., METAXAS D., BADLER N.: User-controlled physics-based animation for articulated figures. In *Proc. of Computer Animation* (1996), vol. 96, Citeseer, pp. 16–25. 9

[KSnL05] KELLY R., SANTIBÁÑEZ V., LORIA A.: *Control of Robot Manipulators in Joint Space (Advanced Textbooks in Control and Signal Processing)*. New York: Springer-Verlag, 2005. 8

[LHP05] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (July 2005), 1071. 5, 8, 16, 19

[LS86] LACQUANITI F., SOECHTING J.: Simulation studies on the control of posture and movement in a multi-jointed limb. *Biological cybernetics 54*, 6 (1986), 367–378. 5

[LvdPF96] LASZLO J., VAN DE PANNE M., FIUME E.: Limit cycle control and its application to the animation of balancing and walking. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (1996), 155–162. 10

[LvdPF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00* (2000), 201–208. 12, 13

[LWZB90] LEE P., WEI S., ZHAO J., BADLER N.: Strength guided motion. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), ACM, p. 262. 5

[LY08] LUENBERGER D., YE Y.: *Linear and nonlinear programming*. Springer Verlag, 2008. 17

[MAEC07] MORIMOTO J., ATKESON C. G., ENDO G., CHENG G.: Improving humanoid locomotive performance with learnt approximated dynamics via Gaussian processes for regression. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 2007), 4234–4240. 15

[McG90] MCGEER T.: Passive dynamic walking. *The International Journal of Robotics Research 9*, 2 (Jan. 1990), 62. 5

[MdLH10] MORDATCH I., DE LASA M., HERTZMANN A.: Robust Physics-Based Locomotion Using Low-Dimensional Planning. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4 (2010), 1–8. 3, 7, 17, 18

[MI97] MUSSA-IVALDI F.: Nonlinear force fields: a distributed

system of control primitives for representing and learning movements. In *The 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA* (1997), IEEE, pp. 84–90. 9

[MLPP09] MUICO U., LEE Y., POPOVIĆ J., POPOVIĆ Z.: Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (July 2009), 1–9. 17, 18

[MWDM98] MATARIC M., WILLIAMSON M., DEMIRIS J., MOHAN A.: Behavior-based primitives for articulated control. In *Proceedings, From Animals to Animats 5, Fifth International Conference on Simulation of Adaptive Behavior* (1998), pp. 165–170. 9

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C. R.: Momentum control for balance. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (2009), 1–8. 4, 17, 18

[MZW99] MATARIĆ M., ZORDAN V., WILLIAMSON M.: Making complex articulated agents dance. *Autonomous Agents and Multi-Agent Systems 2*, 1 (1999), 23–43. 9

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM, pp. 81–88. 2, 9

[NM93] NGO J., MARKS J.: Physically realistic motion synthesis in animation. *Evolutionary Computation 1*, 3 (Sept. 1993), 235–268. 12

[Nov98] NOVACHECK T.: The biomechanics of running. *Gait & Posture 7*, 1 (1998), 77–95. 7

[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. 12

[OM01] OSHITA M., MAKINOUCHI A.: A dynamic motion control technique for human-like articulated figures. *Computer Graphics Forum 20*, 3 (2001), 192–203. 5

[Ott03] OTTEN E.: Inverse and forward dynamics: models of multi-body systems. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences 358*, 1437 (Sept. 2003), 1493–1500. 4

[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. 5, 17

[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 international conference, ICES 2000, Edinburgh, Scotland, April 17-19, 2000, proceedings* (2000), Springer Verlag, p. 175. 15

[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. 1

[PW99] POPOVIĆ Z., WITKIN A.: Physically based motion transformation. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99* (1999), 11–20. 16

[Rai86] RAIBERT M. H.: Legged robots. *Communications of the ACM 29*, 6 (May 1986), 499–514. 9

[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). 4

[RH91] RAIBERT M. H., HODGINS J. K.: Animation of dynamic legged locomotion. *ACM SIGGRAPH Computer Graphics 25*, 4 (July 1991), 349–358. 4, 5, 9, 13

[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. 14, 15, 16

[RM01] REIL T., MASSEY C.: Biologically inspired control of physically simulated bipeds. *Theory in Biosciences 120*, 3-4 (Dec. 2001), 327–339. 15

[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. 13

[SFO05] SHAPIRO A., FALOUTSOS P., OTHERS: Dynamic animation and control environment. *Proceedings of Graphics Interface 2005* (2005), 70. 13

[SH08] SHIRATORI T., HODGINS J. K.: Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Trans. on Graphics (SIGGRAPH Asia) 27*, 5 (2008). 13

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04* (2004), 514. 16

[Sim94] SIMS K.: Evolving virtual creatures. *Proc. of the 21st annual conf. on Computer graphics and interactive techniques - SIGGRAPH '94* (1994), 15–22. 4, 14, 15, 16

[SKL07] SOK K., KIM M., LEE J.: Simulating biped behaviors from human motion data. *ACM SIGGRAPH 2007 papers* (2007), 107. 11, 12, 13, 14

[SKP08] SUEDA S., KAUFMAN A., PAI D. K.: Musculotendon simulation for hand animation. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (2008). 5

[Smi06] SMITH R.: Open Dynamics Engine User Guide v0.5, 2006. 4

[Sta04] STANLEY K.: *Efficient evolution of neural networks through complexification*. The University of Texas at Austin, 2004. 14

[SvdP05] SHARON D., VAN DE PANNE M.: Synthesis of controllers for stylized planar bipedal walking. *Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation, 2005. ICRA 2005* (2005), 2387–2392. 12, 15

[Tag95] TAGA G.: A model of the neuro-musculo-skeletal system for human locomotion. *Biological Cybernetics 73*, 2 (1995), 97–111. 15

[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. 15

[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. 7, 11, 13

[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. 9, 12

[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. 15

[TZS04] TEDRAKE R., ZHANG T., SEUNG H.: Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proceedings of the IEEE international conference on intelligent robots and systems* (2004), pp. 2849–2854. 5, 15

[vdBGEZ07] VAN DEN BOGERT A. J., GEIJTENBEEK T., EVEN-ZOHAR O.: Real-time estimation of muscle forces from inverse dynamics. *health.uottawa.ca* (2007), 5–6. 4

[VDO99] VAUGHAN C., DAVIS B., O'CONNOR J.: *Dynamics of human gait*. Human Kinetics Publishers, 1999. 5

[vdP96] VAN DE PANNE M.: Parameterized gait synthesis. *IEEE Computer Graphics and Applications 16*, 2 (1996), 40–49. 9

[vdP00] VAN DE PANNE M.: Control for simulated human and animal motion. *Annual Reviews in Control* (2000). 15, 16, 19

[vdPF93] VAN DE PANNE M., FIUME E.: Sensor-actuator networks. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), 335 – 342. 14, 15

[vdPKF94] VAN DE PANNE M., KIM R., FIUME E.: Virtual wind-up toys for animation. *Graphics Interface* (1994). 6, 10

[vdPL95] VAN DE PANNE M., LAMOURET A.: Guided optimization for balanced locomotion. *Computer animation and simulation 95* (1995), 165–177. 6

[VL03] VAN DE PANNE M., LEE C.: Ski stunt simulator: Experiments with interactive dynamics. In *Proceedings of the 14th Western Computer Graphics Symposium* (2003). 13

[vWvBE*09] VAN WELBERGEN H., VAN BASTEN B. J. H., EGGES A., RUTTKAY Z., OVERMARS M. H.: Real Time Animation of Virtual Humans: A Trade-off Between Naturalness and Control. *Eurographics - State of the Art Reports* (2009), 45–72. 4

[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. 1

[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. 17

[WFH09] WANG J., FLEET D., HERTZMANN A.: Optimizing Walking Controllers. *ACM Transactions on Graphics (SIGGRAPH Asia)* (2009). 5, 7, 11, 13, 19

[WFH10] WANG J., FLEET D., HERTZMANN A.: Optimizing Walking Controllers for Uncertain Inputs and Environments. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4 (2010), 1–8. 11, 13, 15, 19

[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. 5

[WH00] WOOTEN W., HODGINS J.: Simulating Leaping, Tumbling, Landing, and Balancing Humans. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings* (2000), pp. 656–662. 10

[Wie02] WIEBER P.: On the stability of walking systems. In *Proceedings of the international workshop on humanoid and human friendly robotics* (2002), Citeseer, pp. 53–59. 6

[WJM06] WROTEK P., JENKINS O., MCGUIRE M.: Dynamo: dynamic, data-driven character control with adjustable balance. *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames 1*, 212 (2006), 70. 6

[WK88] WITKIN A., KASS M.: Spacetime constraints. *ACM SIGGRAPH Computer Graphics 22*, 4 (Aug. 1988), 159–168. 16

[WL97] WOODS R., LAWRENCE K.: *Modeling and simulation of dynamic systems*. Prentice Hall Upper Saddle River, New Jersey, 1997. 4

[WP09] WAMPLER K., POPOVIĆ Z.: Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (July 2009), 1. 4

[WP10] WU J.-C., POPOVIC Z.: Terrain-Adaptive Bipedal Locomotion Control. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4 (2010), 1–10. 17, 18

[WZ10] WU C., ZORDAN V.: Goal-Directed Stepping with Momentum Control. *Eurographics / ACM SIGGRAPH Symposium on Computer Animation (2010)* (2010). 18

[Yao99] YAO X.: Evolving artificial neural networks. *Proceedings of the IEEE 87*, 9 (Nov. 1999), 1423–1447. 15

[YCBvdP08] YIN K. K., COROS S., BEAUDOIN P., VAN DE PANNE M.: Continuation methods for adapting simulated skills. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (2008), 1–7. 11, 13, 19

[YCP03] YIN K. K., CLINE M. B., PAI D. K.: Motion perturbation based on simple neuromotor control models. *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings* (2003), 445–449. 4, 12

[YL08] YE Y., LIU C. K.: Animating responsive characters with dynamic constraints in near-unactuated coordinates. *ACM Transactions on Graphics (SIGGRAPH Asia) 27*, 5 (Dec. 2008), 1. 19

[YLS04] YANG P., LASZLO J., SINGH K.: Layered dynamic control for interactive character swimming. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), Eurographics Association Aire-la-Ville, Switzerland, Switzerland, pp. 39–47. 12

[YLvdP07] YIN K. K., LOKEN K., VAN DE PANNE M.: Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (SIGGRAPH) 26*, 3 (2007), 105. 5, 10, 11, 12, 13, 14

[ZH99] ZORDAN V., HODGINS J.: Tracking and modifying upper-body human motion data with dynamic simulation. *Computer Animation and Simulation 2* (1999). 11

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '02* (2002), 89. 9, 11, 12, 13

[ZMCM05] ZORDAN V., MAJKOWSKA A., CHIU B., M: Dynamic response for motion capture animation. *ACM Transactions on Graphics (TOG) 1*, 212 (2005), 697–701. 19

[ZSC90] ZATSIORSKY V., SELUYANOV V., CHUGUNOVA L.: Methods of determining mass-inertial characteristics of human body segments. *Contemporary problems of biomechanics* (1990), 272–291. 5

[ZvdP05] ZHAO P., VAN DE PANNE M.: User interfaces for interactive control of physics-based 3d characters. *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), 87–94. 13