Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

Simple Data-Driven Control for Simulated Bipeds

T. Geijtenbeek N. Pronost A. F. van der Stappen

Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

We present a framework for controlling physics-based bipeds in a simulated environment, based on a variety of reference motions. Unlike existing methods for control based on reference motions, our framework does not require preprocessing of the reference motion, nor does it rely on inverse dynamics or on-line optimization methods for torque computation. It consists of three components: Proportional-Derivative Control *to mimic motion characteristics, a specific form of* Jacobian Transpose Control *for balance control, and* Covariance Matrix Adaption *for off-line parameter optimization, based on a novel high-level reward function. The framework can easily be implemented using common off-the-shelf physics engines, and generates simulations at approximately* 4× *real-time on a single core of a modern PC. Our framework advances the state-of-the-art by demonstrating motions of a diversity and dynamic nature previously unseen in comparable methods, including squatting, bowing, kicking, and dancing motions. We also demonstrate its ability to withstand external perturbations and adapt to changes in character morphology.*

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

After decades of floundering, recent publications show that the field of physics-based character animation has matured, demonstrating a visual quality and robustness that can compete with kinematics-based approaches. However, despite widespread adoption of physics simulation for lifeless phenomena, such as cloth, water or *rag-doll* characters, production games still resort to kinematics-based approaches for the animation of actively controlled characters.

In an attempt to understand this reservation, we distinguish between two different approaches used in physics-based character animation research. On one hand, there are methods that compute joint torques based on *kinematic parameters*, such as joint angles or center-of-mass. These methods are relatively easy to implement and integrate well with common physics engines, such as the *Open Dynamics Engine* or *PhysX*. Recent publications that use this approach have displayed a variety of robust and versatile locomotion controllers (*e.g.* [CBvdP10, WFH10]). However, these methods allow style control only through *key-framing* or optimization for high-level criteria. Methods that use captured reference motions have several limitations or require extensive preprocessing (*e.g.* [SKL07, YLvdP07]).

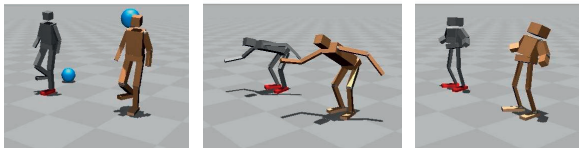

Figure 1: *Example results of the control framework.*

On the other hand, there are methods that compute joint torques based on the *equations-of-motion* that describe the underlying character dynamics. Such methods have demonstrated impressive results, including the ability to robustly track recorded running and turning motions, as well as robust navigation over uneven terrain (e.g. [MLPP09, WP10, MdLH10]). However, these methods are also much more difficult to implement. They require thorough knowledge of constrained dynamics and optimization theory and do not mix trivially with common physics engines. In addition, computational requirements are often high: publications typically report around 1× real-time performance for a single character on a modern PC.

© The Eurographics Association 2012.

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

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

2. Related Work

Following our introduction, we distinguish between publications that compute joint torques based on *kinematic parameters* and those that compute torques based on the *equations-of-motion* describing the underlying character dynamics.

Torques Computed from Kinematic Properties Early examples using this approach include the work of Raibert and Hodgins [RH91], as well as the milestone human athletics controller by Hodgins et al. [HWBO95]. Both techniques control speed and balance through on-line modulation of target trajectories, and compute torques using Proportional-Derivative Control. Other key examples are the walking controller of Laszlo et al. [LvdPF96], the various controllers of and Faloutsos et al. [FvdPT01], and the interactive tracking controller of Zordan and Hodgins [ZH02]. More recently, Yin et al. [YLvdP07] introduced SIMBICON, a framework supporting various types of robust gaits, also using on-line trajectory modulation for balance and PD Control, which has been the basis for many subsequent publications [CBYvdP08, CBvdP09, YCBvdP08, WFH09, WFH10]. Coros et al. incorporate a form of *Jacobian Transpose Control* [SADM94], resulting in versatile and generic locomotion controllers for bipeds [CBvdP10] and quadrupeds [CKJ*11]. Their control method is based on the *Virtual Model Control* framework by Pratt et al. [PCTD01] (see also Section 3.2.1).

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

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

Torques Computed from Equations-of-Motion This category of approaches is linked to the *spacetime* animation framework by Witkin and Kass [WK88] and computes torques through constrained optimization based on the equations-of-motions describing the character dynamics. Stewart and Cremer [SC92] use this method to produce physically correct animations for climbing and descending stairs. Abe et al. [AdSP07] use quadratic programming (QP) to find the set of torques that optimally drive the center-of-mass over the base of support, demonstrating robust balance behaviors on moving bases. Macchietto et al. [MZS09] use inverse dynamics to track trajectories that minimize angular momentum, to which Wu and Zordan [WZ10] add stepping motions for additional balance correction. Lee et al. [LKL10] use inverse dynamics in combination with a SIMBICON-like balance strategy to robustly track captured reference motions. De Lasa et al. [dLMH10] use on-line prioritized optimization to construct robust locomotion and jumping controllers. Muico et al. [MLPP09, MPP11] use off-line optimization of reference motions and contact forces, in combination with a nonlinear quadratic regulator to create agile walking and running controllers. Wu and Popović [WP10] use optimized end-effector trajectories in combination with QP for navigation over uneven terrain.

Other publications use short-horizon optimization of an internal model to acquire an on-line look-ahead policy. Examples are the work of Da Silva et al. [dSAP08], who demonstrate walking controllers based on reference motions, and Kwon and Hodgins [KH10], who demonstrate running controllers based on reference motions. Mordatch et al. [MdLH10] demonstrate locomotion over constrained and uneven terrain, using the low-level control framework

© The Eurographics Association 2012.

of [dLMH10]. Jain et al. [JL11] perform motion tracking using a model based on the principal modes of the character, while Ye and Liu [YL10] use an abstract internal model based on center-of-mass and ground reaction force.

Because of the tight link between control and simulation in these methods, physics simulation is mostly performed as part of the control framework. Exceptions of publications in which such methods have been integrated with common physics engines are the work of Da Silva et al. [dSAP08] (*Open Dynamics Engine*) and the uneven terrain controller of Wu and Popović [WP10] (*PhysX*).

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

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

The main difference between our work and that of Zordan and Hodgins [ZH02] is that their controllers are limited to dual stance motions with upper-body dynamics, while we demonstrate motions of both single and double stance, with significant lower-body dynamics. The main difference between our work and that of Coros et al. [CBvdP10] is that in their framework, motion trajectories are constructed through key poses, while our framework uses unaltered reference motion trajectories. An important difference between our work and that of Lee et al. [LKL10] is that their method requires inverse dynamics for torque computation. Inverse dynamics is not supported by most common physics engines and has additional computational requirements. In addition, inverse dynamics control inefficiently fights the natural dynamics of a physics-based character [PCTD01], which may lead to unnatural joint compliance.

Another key difference between our work and that of both Coros et al. [CBvdP10] and Lee et al. [LKL10] is that their balance strategies are largely based on swing foot placement for balance – inspired by the SIMBICON balance strategy of Yin et al. [YLvdP07]. This makes their control frameworks suitable for locomotion, while the static balance algorithms used in our framework make it more suitable for in-place motions.

3. Control Framework

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

- *Tracking control* (Section 3.1). To mimic the characteristics of a recorded motion, we track the reference trajectories of the individual degrees of freedom (DOFs) using Proportional-Derivative (PD) Control.
- *Balance control* (Section 3.2). To maintain balance, we use a specific form of Jacobian Transpose (JT) Control, consisting of a set of *virtual forces* and *virtual torques*.
- *Off-line parameter optimization* (Section 3.3) To find the right set of control parameters for a specific motion or character, we perform off-line parameter optimization based on high-level objectives, using Covariance Matrix Adaption (CMA).

At any time t, we assume availability of the values of all degrees of freedom and their first order derivatives, both for a simulated character, C, and a reference motion, A. For a simulated character, DOFs and their derivatives are accessible from the physics simulation engine. For a reference motion, the first order derivatives can be acquired through low-pass filtering and differentiation.

3.1. Tracking Control

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

$$\tau_{pd} = k_p(\theta_A - \theta_C) + k_d(\dot{\theta}_A - \dot{\theta}_C) \quad (1)$$

The responsiveness to deviations in position and velocity is controlled through gains k_p and k_d, which are determined individually for each actuated DOF through off-line optimization (see Section 3.3).

3.2. Balance Control

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

© The Eurographics Association 2012.

3.2.1. Jacobian Transpose Control

The essence of this control method is that it enables control in Cartesian space for linked structures with redundant DOFs. Using the Jacobian Transpose, it is possible to compute the set of torques that emulate the effect of an external force or torque, applied to a specific body or a virtual point (such as the center-of-mass). Virtual forces and torques are applied to a *chain of linked bodies*, starting from a static base link (such as the stance foot) and moving to a target link (see Figure 2 for an illustration). The set of joint torques τ_F that emulate a virtual force F applied at point p corresponds to:

$$\tau_F = J(p)^T F \tag{2}$$

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

$$J(p)^T = \begin{bmatrix} \frac{\partial p_x}{\partial \alpha_1} & \frac{\partial p_y}{\partial \alpha_1} & \frac{\partial p_z}{\partial \alpha_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial p_x}{\partial \alpha_k} & \frac{\partial p_y}{\partial \alpha_k} & \frac{\partial p_z}{\partial \alpha_k} \end{bmatrix} \tag{3}$$

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

$$J(p)_i^T = \begin{bmatrix} \frac{\partial p_x}{\partial \alpha_i} & \frac{\partial p_y}{\partial \alpha_i} & \frac{\partial p_z}{\partial \alpha_i} \end{bmatrix} = (a_i \times (p - b_i))^T \tag{4}$$

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

$$\tau_{F,i} = (a_i \times (p - b_i))^T F \tag{5}$$

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

$$J(e)_i^T = \begin{bmatrix} \frac{\partial e_x}{\partial \alpha_i} & \frac{\partial e_y}{\partial \alpha_i} & \frac{\partial e_z}{\partial \alpha_i} \end{bmatrix} = a_i^T \tag{6}$$

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

$$\tau_{T,i} = a_i^T T \tag{7}$$

Note that, since bipeds contain no links that are truly static, the effect of a virtual force or torque is always an approximation and can be determined only after simulation. That said, we have found this approximation to be sufficiently accurate to work well in practice.

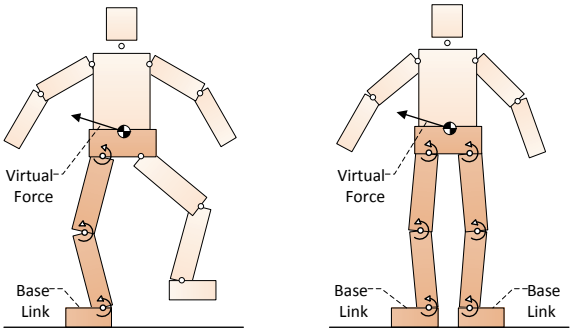

Figure 2: *Virtual forces applied to the center of mass, through a chain of linked bodies, for single stance (left) and dual stance (right).*

3.2.2. Stance State

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

$$S \in \{\text{left\_stance}, \text{right\_stance}, \text{dual\_stance}, \text{flight}\} \tag{8}$$

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

3.2.3. Base of Support

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

$$p_{\text{base}} = \begin{cases} \perp p_{\text{ankL}} & \text{if } S_A = \text{left\_stance} \\ \perp p_{\text{ankR}} & \text{if } S_A = \text{right\_stance} \\ \frac{\perp p_{\text{ankL}} + \perp p_{\text{ankR}}}{2} & \text{otherwise} \end{cases} \tag{9}$$

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

3.2.4. Balance Strategy Components

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

- A virtual force applied to the center-of-mass (COM) of the character, to minimize differences in COM position and velocity between the simulated character and the reference motion.

© The Eurographics Association 2012.

- A virtual torque applied to the pelvis, to maintain the upper-body posture of the simulated character.
- A virtual torque applied to the pelvis, to regulate the total angular momentum of the character.

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

Center-of-Mass Position and Velocity The first component of our balance control strategy compensates for differences in COM position and velocity. It does so by applying a virtual force at the COM position of the simulated character, to the chain of bodies from stance foot to pelvis (see Figure 2). For a character consisting of k bodies, COM position p_{com} and COM velocity v_{com} are a weighted average of individual body positions, $p_{\text{body},i}$, and velocities $v_{\text{body},i}$:

$$p_{\text{com}} = \sum_{i=1}^{k} \frac{\text{m}_i}{\text{m}_{\text{tot}}} p_{\text{body},i} \quad , \quad v_{\text{com}} = \sum_{i=1}^{k} \frac{\text{m}_i}{\text{m}_{\text{tot}}} v_{\text{body},i} \quad (10)$$

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

We attempt to control balance by regarding the COM position of a character with respect to its base of support:

$$\hat{p}_{\text{com}} = p_{\text{com}} - p_{\text{base}} \quad (11)$$

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

$$F_{\text{com}} = \text{w}_{\text{cp}}(\hat{p}_{\text{com},A} - \hat{p}_{\text{com},C}) + \text{w}_{\text{cv}}(v_{\text{com},A} - v_{\text{com},C}) \quad (12)$$

in which w_{cp} and w_{cv} are constants controlling force magnitude. These constants are determined through off-line optimization (see Section 3.3). Different parameter sets are used depending on the stance state of the character (single stance or dual stance). The corresponding set of individual joint torques, τ_{com}, which are applied to each DOF in *stance ankle*, *stance knee* and *stance hip* can be acquired using Equation (5).

Trunk Orientation To maintain the posture of the upper body, we attempt to minimize the difference in trunk orientation. We do so by applying a virtual torque to the chain of bodies from stance foot to the pelvis (see Figure 2). If q_A is a quaternion describing the trunk orientation of the reference motion, and q_C that of the simulated character, then the virtual torque T_{trunk} corresponds to:

$$T_{\text{trunk},i} = \text{w}_{\text{to}}\ expmap(q_{\text{trunk},C}^{-1}\, q_{\text{trunk},A}) \quad (13)$$

where $expmap : \mathbb{R}^4 \Rightarrow \mathbb{R}^3$ is a function that extracts an *exponential map* from a quaternion (see [Gra98]), and w_{to} is a constant controlling the magnitude of the torque, determined through off-line optimization (see Section 3.3). The corresponding set of individual joint torques, τ_{trunk}, which are applied to each DOF in *stance ankle*, *stance knee* and *stance hip* can be acquired through Equation (7).

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

$$L = \sum_{i=1}^{k} \text{m}_i(p_{\text{body},i} - p_{\text{com}}) \times v_{\text{body},i} + \text{R}_{\text{world},i}\text{I}_i\omega_i \quad (14)$$

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

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

$$T_{\text{am}} = \text{w}_{\text{am}}(L_A - L_C) \quad (15)$$

in which w_{am} is a constant controlling magnitude, determined through off-line optimization (see Section 3.3). The corresponding set of individual joint torques, τ_{am}, which are applied to each DOF in *stance ankle*, *stance knee* and *stance hip* can be acquired through Equation (7).

3.2.5. Combining the Individual Components

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

$$\tau_{\text{control}} = \tau_{\text{pd}} + \tau_{\text{com}} + \tau_{\text{trunk}} + \tau_{\text{am}} \quad (16)$$

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

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

$$\tau_i = \begin{cases} \tau_{\text{max},i} + \tau_{\text{noise},i} & \text{if } \tau_{\text{control},i} > \tau_{\text{max},i} \\ -\tau_{\text{max},i} + \tau_{\text{noise},i} & \text{if } \tau_{\text{control},i} < -\tau_{\text{max},i} \\ \tau_{\text{control},i} + \tau_{\text{noise},i} & \text{otherwise} \end{cases} \quad (17)$$

© The Eurographics Association 2012.

3.3. Off-line Parameter Optimization

In this final step of our method, we optimize the various parameters of our control framework, based on high-level optimization criteria. The set of parameters we wish to optimize consists of the *PD Controller gains* and the *weights from our balance control strategy*:

$$\mathrm{K} = \left\{ \mathrm{k}_{\mathrm{p},1}, \ldots, \mathrm{k}_{\mathrm{p},n}\ ,\ \mathrm{k}_{\mathrm{d},1}, \ldots, \mathrm{k}_{\mathrm{d},n} \right\} \quad (18)$$

$$\mathrm{W}_{\mathrm{single}} = \mathrm{W}_{\mathrm{dual}} = \left\{ \mathrm{w}_{\mathrm{cv}}, \mathrm{w}_{\mathrm{cp}}, \mathrm{w}_{\mathrm{to}}, \mathrm{w}_{\mathrm{am}} \right\} \quad (19)$$

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

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

$$\mathrm{P} = \left\{ \mathrm{K}, \mathrm{W}_{\mathrm{single}}, \mathrm{W}_{\mathrm{dual}} \right\} \quad , \quad \|\mathrm{P}\| = 2n + 8 \quad (20)$$

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

- *Pose displacement* (e_{pose}). The pose of the simulated character should not deviate too much from the reference motion. An evident case in which this occurs is when the simulated biped has lost its balance. We define pose displacement, $e_{\mathrm{pose}}(t) \rightarrow \mathbb{R}$, as the weighted average of the displacement of the individual bodies, with respect to the COM of the character.
- *Stance state error* (e_{stance}). Sometimes the stance state of the simulated character is different from that of the reference motion, *e.g.* S_A = single_stance while S_C = dual_stance. Such a difference is undesireable, and can occur even during small pose displacements. We define the contact state error, $e_{\mathrm{stance}}(t) \rightarrow [0,1]$, as the ratio of time during which $S_A \neq S_C$, in the window between t and $t-2$ (or t and 0 when $t < 2$).
- *Foot sliding* (e_{slide}). The feet of a simulated character sometimes slide as a result of a specific combination of internal joint torques, while the feet of the reference motion stand firmly. Since we measure body displacement relative to the COM position, this sliding often does not lead to significant errors in pose displacement. We define foot sliding error, $e_{\mathrm{slide}}(t) \rightarrow \mathbb{R}$, as the average speed in the horizontal plane of the stance foot, during the time window between t and $t-2$ (or t and 0 when $t < 2$). During dual stance, we use the sum of both feet.
- *Torque* (e_{torque}). Finally, we wish to control the amount of torque used by the character's actuators. We define joint torque, $e_{\mathrm{torque}}(t) \rightarrow \mathbb{R}$, as the average summed torque of all actuated DOFs, during the time window between t and $t-2$ (or t and 0 when $t < 2$).

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

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

$$R(\mathrm{P}) = \frac{t_{\mathrm{term}}}{t_{\mathrm{max}}} + \mathrm{w}_{\mathrm{bonus}} \frac{t_{\mathrm{term}}}{t_{\mathrm{max}}} \frac{1}{\|\mathrm{E}\|} \sum_{e \in \mathrm{E}} \left[1 - \frac{e_{avg}}{e_{max}} \right] \quad (21)$$

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

Optimization Strategy Our fitness landscape is irregular, with interdependence between parameters and many local maximums. Following recent publications in physics-based control [WFH09, WFH10, WP10, TGTL11], we use Covariance Matrix Adaption (CMA) [Han06] for off-line parameter optimization. CMA is an evolutionary strategy that attempts to learn the *covariance matrix* of the current region of the fitness landscape through random sampling. Many freeware implementations of CMA exist, including the Shark library [IGHM08].

4. Experimentation

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

Setup For our experimentation, we have selected a set of 10 different reference motions (see Table 1). The motions

© The Eurographics Association 2012.

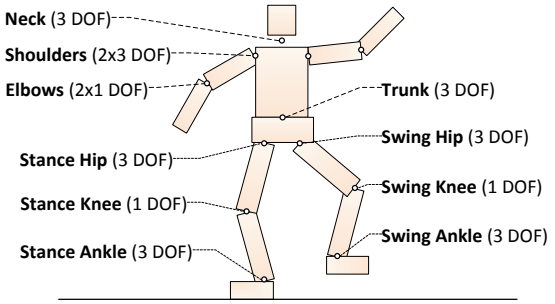

Figure 3: *Character Degrees-of-Freedom.*

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

Table 1: *List of the motion clips used in experimentation.*

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

Table 2: *Overview of the 42 parameters used in off-line optimization (the sum of Dim $\times$ Sets). iMin and iMax indicate lower and upper ranges used for random initialization.*

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

Table 3: *Overview of optimization performance and resistance to external perturbations. 'Gen' is the number of generations required for optimization; 'Time' is the estimated optimization time, based on 4 $\times$ real-time performance.*

were captured using an 8 camera Vicon system, fitting a 28 DOF character into a marker setup consisting of 41 markers (see Figure 3). We filtered the resulting DOF trajectories using a real-time 2nd order Butterworth filter with a cut-off frequency of 3Hz (for the 'dance' motion we use a cut-off frequency of 2Hz).

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

Optimization For optimization, we use $e_{\text{pose,max}} = 0.1$, $e_{\text{stance,max}} = 0.5$, $e_{\text{slide,max}} = 0.25$, $e_{\text{torque,max}} = 1000$ and $t_{\text{term}} = 20$ (for the 'dance' motion we use $e_{\text{stance,max}} = 1.0$ and $e_{\text{slide,max}} = 0.35$). For the CMA algorithm, we use $\lambda = 16$ and $\mu = 8$ (see [Han06] for details). See Table 2 for an overview of the parameters used in optimization, as well as their initialization settings and range. We declare the optimization of the control parameters a success after a threshold of $R(\text{P}) = 1.8$ has been reached. The number of generations before success, as well as the estimated optimization time are displayed in Table 3.

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

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

Changes in Character Morphology Another advantage of physics-based character animation is the ability of con-

© The Eurographics Association 2012.

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

Table 4: *Ability of controllers to be used with different motions. Crosses indicate the ability of a controller optimized for the motion on the left to be reused for the motion on top, without additional optimization.*

trollers to adapt to changes in character morphology, while maintaining physical correctness. To demonstrate this, we optimize controllers for characters with different body dimensions, using the same set of reference motion trajectories. One tested morphology contained a short and heavy upper-body, with thin legs; another contained a long upper-body with long arms (see also Figure 1). In our balance control algorithm, we moved the reference target COP above the base of support and set the reference angular momentum to zero ($L_A = 0$), since the original values no longer represent physically valid motion. We have found that, with the exception of 'kick' and 'side', our control framework was able to adapt to these changes in character morphology, producing motions that fitted the new morphologies characteristically (see also the the supplementary video material).

Reusing Controllers We also tested the capability of control parameters optimized for one motion to be used for other motions. Table 4 displays the results of these experiments. To increase robustness, the parameters were optimized using maximum push force perturbation.

5. Discussion and Future Work

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

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

Our framework relies on off-line optimization of its control parameters, and it can be argued that this is similar to requiring preprocessing of reference motion trajectories. However, as we have shown, controller settings of our framework can be reused for different motions without the need for additional optimization. We expect that it will be possible to construct a database of parameter settings that can be used to automatically select the right set of parameters for a given character and type of motion, without the need for additional optimization.

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

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

Torque Minimization Another angle we wish to investigate further is the effect of torque minimization on perturbation response. A lower value of $e_{\text{torque,max}}$ (or a differently formulated error term) may promote less stiff behavior and lead to more natural compliance.

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

References

[AdSP07] ABE Y., DA SILVA M., POPOVIĆ J.: Multiobjective control with frictional contacts. *ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2007), 249–258. 2

[CBvdP09] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust Task-based Control Policies for Physics-based Characters. *ACM Transactions on Graphics 28*, 5 (2009). 2

© The Eurographics Association 2012.

[CBvdP10] COROS S., BEAUDOIN P., VAN DE PANNE M.: Generalized biped walking control. *ACM Transactions on Graphics 29* (2010). 1, 2, 3, 8

[CBYvdP08] COROS S., BEAUDOIN P., YIN K. K., VAN DE PANNE M.: Synthesis of constrained walking skills. *ACM Transactions on Graphics 27*, 5 (Dec. 2008). 2

[CKJ*11] COROS S., KARPATHY A., JONES B., REVERET L., VAN DE PANNE M.: Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics 30*, 4 (2011). 2

[dLMH10] DE LASA M., MORDATCH I., HERTZMANN A.: Feature-Based Locomotion Controllers. *ACM Transactions on Graphics 29*, 3 (2010). 2, 3

[dSAP08] 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. 2, 3

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *ACM SIGGRAPH Papers* (2001), pp. 251–260. 2

[Gra98] GRASSIA F. S.: Practical Parameterization of Rotations Using the Exponential Map. *The Journal of Graphics Tools 3*, 3 (1998), 1–13. 4, 5

[Han06] HANSEN N.: The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation* (2006), 75–102. 2, 6, 7

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated behaviors for new characters. In *ACM SIGGRAPH Papers* (1997), pp. 153–162. 2

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *ACM SIGGRAPH Papers* (1995), pp. 71–78. 2

[IGHM08] IGEL C., GLASMACHERS T., HEIDRICH-MEISNER V.: Shark. *Journal of Machine Learning Research 9* (2008), 993–996. 6

[Jaz10] JAZAR R.: *Theory of applied robotics: kinematics, dynamics, and control*. Springer Verlag, 2010. 4

[JL11] JAIN S., LIU C.: Modal-space control for articulated characters. *ACM Transactions on Graphics 30*, 5 (2011). 3

[KH10] KWON T., HODGINS J.: Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2010), pp. 129–138. 2

[LKL10] LEE Y., KIM S., LEE J.: Data-driven biped control. *ACM Transactions on Graphics 29*, 4 (July 2010), 129. 2, 3, 8

[LvdPF96] LASZLO J., VAN DE PANNE M., FIUME E.: Limit cycle control and its application to the animation of balancing and walking. In *ACM SIGGRAPH Papers* (1996), pp. 155–162. 2

[MdLH10] MORDATCH I., DE LASA M., HERTZMANN A.: Robust Physics-Based Locomotion Using Low-Dimensional Planning. *ACM Transactions on Graphics 29*, 4 (2010). 1, 2

[MLPP09] MUICO U., LEE Y., POPOVIĆ J., POPOVIĆ Z.: Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28*, 3 (July 2009). 1, 2

[MPP11] MUICO U., POPOVIĆ J., POPOVIĆ Z.: Composite control of physically simulated characters. *ACM Transactions on Graphics 30*, 3 (May 2011). 2

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C. R.: Momentum control for balance. *ACM Transactions on Graphics 28*, 3 (2009). 2, 5

[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. 2, 3

[RH91] RAIBERT M. H., HODGINS J. K.: Animation of dynamic legged locomotion. *ACM SIGGRAPH Computer Graphics 25*, 4 (July 1991), 349–358. 2

[SADM94] SUNADA C., ARGAEZ D., DUBOWSKY S., MAVROIDIS C.: A coordinated Jacobian transpose control for mobile multi-limbed robotic systems. In *IEEE Int. Conf. on Robotics and Automation* (1994), pp. 1910–1915. 2

[SC92] STEWART A., CREMER J.: Beyond keyframing: An algorithmic approach to animation. In *Graphics Interface* (1992), pp. 273–281. 2

[Sim94] SIMS K.: Evolving virtual creatures. In *ACM SIGGRAPH Papers* (1994), pp. 15–22. 2

[SKL07] SOK K., KIM M., LEE J.: Simulating biped behaviors from human motion data. *ACM Transactions on Graphics 26*, 3 (2007), 107. 1, 2, 3

[Smi06] SMITH R.: Open Dynamics Engine User Guide v0.5, 2006. 7

[SvdP05] SHARON D., VAN DE PANNE M.: Synthesis of controllers for stylized planar bipedal walking. *Proc. of the Int. Conf. on Robotics and Automation* (2005), 2387–2392. 2

[TGTL11] TAN J., GU Y., TURK G., LIU C.: Articulated swimming creatures. *ACM Transactions on Graphics 30*, 4 (2011), 58. 2, 6

[WFH09] WANG J., FLEET D., HERTZMANN A.: Optimizing walking controllers. *ACM Transactions on Graphics 28*, 5 (2009), 168. 2, 6

[WFH10] WANG J., FLEET D., HERTZMANN A.: Optimizing Walking Controllers for Uncertain Inputs and Environments. *ACM Transactions on Graphics 29*, 4 (2010), 1–8. 1, 2, 5, 6

[WH00] WOOTEN W., HODGINS J.: Simulating leaping, tumbling, landing and balancing humans. In *IEEE Int. Conf. on Robotics and Automation* (2000), vol. 1, IEEE, pp. 656–662. 2

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *ACM SIGGRAPH Computer Graphics* (Aug. 1988), ACM, pp. 159–168. 2

[WP10] WU J.-C., POPOVIC Z.: Terrain-Adaptive Bipedal Locomotion Control. *ACM Transactions on Graphics 29*, 4 (2010). 1, 2, 3, 6

[WZ10] WU C., ZORDAN V.: Goal-Directed Stepping with Momentum Control. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2010), Citeseer, pp. 113–118. 2

[YCBvdP08] YIN K. K., COROS S., BEAUDOIN P., VAN DE PANNE M.: Continuation methods for adapting simulated skills. *ACM Transactions on Graphics 27*, 3 (2008). 2

[YH09] YAMANE K., HODGINS J.: Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data. In *Intelligent Robots and Systems* (Oct. 2009), IEEE, pp. 2510–2517. 8

[YL10] YE Y., LIU C.: Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics 29*, 4 (July 2010), 74. 3

[YLvdP07] YIN K. K., LOKEN K., VAN DE PANNE M.: Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (2007), 105. 1, 2, 3, 8

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2002), ACM Press, p. 89. 2, 3

© The Eurographics Association 2012.