Trajectory Optimization for Full-Body Movements with Complex Contacts

Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann, *Senior Member*, *IEEE*

Abstract—This paper presents the first method for full-body trajectory optimization of physics-based human motion that does not rely on motion capture, specified key-poses, or periodic motion. Optimization is performed using a small set of simple goals, for example, one hand should be on the ground, or the center-of-mass should be above a particular height. These objectives are applied to short spacetime windows which can be composed to express goals over an entire animation. Specific contact locations needed to achieve objectives are not required by our method. We show that the method can synthesize many different kinds of movement, including walking, hand walking, breakdancing, flips, and crawling. Most of these movements have never been previously synthesized by physics-based methods.

Index Terms—Physics-based animation, character animation, motion control

1 Introduction

It has long been hypothesized that many human motions can be described as optimizing simple objective functions based on task goals and energy consumption. Optimality is appealing as both a simple explanation for how we move, and a simple parameterization for motion. For character animation, optimality could allow the creation of many types of natural human movements from high-level specifications, without requiring labor-intensive keyframing or motion capture. Optimization of a movement is called trajectory optimization, or, equivalently, spacetime optimization.

Unfortunately, progress on full-body trajectory optimization has been limited, because the problem is very high-dimensional and objectives are highly nonlinear, which often lead optimization procedures into bad local minima. The best existing methods rely on motion capture data, which limits their generality, apply only to periodic locomotion, or use simplified character models (such as planar models with point feet) that eliminate important degrees-of-freedom. Despite recent progress in physics-based character animation, full-body trajectory optimization remains an open problem.

This paper presents a method for full-body optimization of human movement. Optimization is performed using a small set of simple goals, for example, specifying that a hand should be on the ground or that the center-of-mass (COM) should be above a particular height. These objectives are applied to short spacetime windows, which can be composed to express goals over an entire animation, such as achieving standing balance or locomotion. Optimization is performed using covariance matrix adaptation.

The main technical contribution of this work is in the design of a set of modular and reusable objective functions for many types of full-body motion. We show that just a few, easily interpretable—but carefully chosen—objectives are sufficient to describe and create many diverse types of motion, including walking, exercise, and breakdancing maneuvers. These skills include low- and high-energy motions, as well as highly rotational behaviors. Adjusting constraint parameters yields motions that are more superhuman or more typical of normal human movements. All motions reuse the same straightforward initialization procedure. To our knowledge, many of the presented motions have never been successfully synthesized by previous trajectory optimization methods.

Our work focuses on the problem of generating full-body motion, given a high-level plan, namely the sequence of spacetime windows. In this paper, we specify this sequence by hand and leave higher level plan generation as future work. In many cases, results do not perfectly match natural human motion. This may be due to the simplified nature of our musculoskeletal model and lack of interlimb contact; more accurate models [1] could help improve results. Though our optimizations are expensive, they demonstrate for the first time the feasibility of trajectory optimization for creation of a diverse set of whole-body motions. Future research could focus on accelerating the optimizations and improving realism.

2 Related Work

An open question in computer graphics is how to automatically synthesize physics-based character animation. To date, two main approaches have emerged to tackle this problem, trajectory optimization and controllers.

- *M. Al Borno is with the Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada. E-mail: mazen@dgp.toronto.edu.*
- *M. de Lasa is with Autodesk Canada, 210 King St. East, Toronto, ON M5A1J7, Canada. E-mail: mdelasa@dgp.toronto.edu.*
- *A. Hertzmann is with Adobe Systems and the Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada. E-mail: hertzman@dgp.toronto.edu.*

Manuscript received 18 Sept. 2012; revised 8 Dec. 2012; accepted 11 Dec. 2012; published online 21 Dec. 2012.
Recommended for acceptance by R. Boulic.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-09-0203.
Digital Object Identifier no. 10.1109/TVCG.2012.325.

Trajectory optimization or spacetime constraints [2] poses motion synthesis as a constrained nonlinear optimization. Early motion synthesis work focused on examples with a small number of joints and used key poses to direct animation [3], [4]. To broaden the class of movements, several approaches focused on highly dynamic actions, dominated by a small set of constraints [5], [6]. With the exception of challenging balancing scenarios [7], generalizing these methods to broader classes of full-body 3D characters has proven difficult due to the high dimensional, nonlinear, nature of these problems. Defining good objectives for low-energy motions such as walking has also proven to be difficult.

Recent work has used motion capture data to ease authoring of new motions. Several editing methods have been proposed that adapt recordings to new situations, while maintaining physical properties [8], [9], [10], [11]. Safonova et al. [12] learn low-dimensional bases from motion data, thereby making optimization easier and capturing stylistic aspects of motion. Liu et al. [13] learn objective function parameters from motion data. Sok et al. [14] and Liu et al. [15] adapt reference input motions using randomized search. Such strategies have produced compelling results with many lifelike qualities. However, their output is limited to remain near provided input data. Our work requires neither good initialization nor motion capture to synthesize complex, contact-rich motions. The method can be used for both highly dynamic behaviors, such as flips and spins, as well as less energetic motions, including walking.

To keep problems tractable and limit the search space, a number of methods focus on cyclic behaviors. Wampler and Popović [16] focus on periodic motions, for simplified characters with point feet, and optimize morphology along with gait parameters. Nunes et al. [17] use a periodic trajectory representation to generate a variety of running behaviors. Others focus on behaviors without contact, such as flying [18], and swimming [19], which further simplifies the optimization landscape. Closely related to our work, Mordatch et al. [20] simulatenously optimize for contact and behavior, for motions with complex contacts including climbing and cooperative motions. Their method currently optimizes motions according to a simplified physics model, uses a "one-way" contact model, and does not handle highly dynamic motion.

As with several of these works, we also employ a shooting strategy [21], which means that our method can be applied to off-the-shelf dynamics simulators and requires no computation of derivatives. Our approach works for a wide variety of cyclic and acyclic movements, allows composition of long motion sequences, and does not require any special treatment of complex contacts. Although we focus on human motion, our work can be easily extended to other types of creatures.

An alternative to trajectory optimization is to use action-specific controllers. This approach has been used to synthesize numerous locomotion and athletic behaviors [22], [23], [24], [25] with recent methods focused on low-dimensional parameterizations [26], [27], [28], [29], [30], [31] and robust navigation of uneven terrain [32], [33]. Our method is complementary to this work. Open-loop movements generated by our system could be used as the basis for controllers or libraries of control primitives. This would enable reuse of results from our system in interactive applications. The proposed look-ahead strategy is also closely related to model-predictive methods that plan over short windows into the future [32], [34], [35], [36].

The objectives we propose could also be used by methods that use optimization to tune settings for predefined controllers [28], [37], [1], [38]. Though we generate open-loop behaviors and leave addition of feedback for future work, our method requires no design of action-specific controller before new motions can be generated.

3 Parameterization and Optimization

We optimize full-body character motion in a physically simulated environment, using only a small set of high-level objectives. Goals for the motion are expressed in terms of an objective function E, with energy terms E_i for motion properties such as stride length and angular momentum. The specific terms of the energy depend on the type of desired motion.

We use a character with 41 degrees-of-freedom in the kinematic pose $\mathbf{q}$, and represent a motion as time-varying poses $\mathbf{q}_{1:T}$. Objectives E_i are combined by weighted combination with weights w_i:

$$E(\mathbf{q}_{1:T}) = \sum_i w_i E_i(\mathbf{q}_{1:T}). \tag{1}$$

Trajectory optimization applied directly to joint torques or to poses is very difficult, due to the highly discontinuous relationship between these quantities and task completion. For this reason, we parameterize the motion in terms of a reference trajectory [19]. The reference trajectory $\hat{\mathbf{q}}_{1:T}$ is represented as a cubic B-spline, with C_0 continuity at key-poses $\mathbf{x}$. The free variables in optimization are the key-poses of the spline.

Given a reference trajectory

$$\hat{\mathbf{q}}_{1:T} = \mathbf{B}\mathbf{x}, \tag{2}$$

where $\mathbf{B}$ is a spline basis matrix, the output motion $\mathbf{q}_{1:T}$ is computed by simulation. At each time index t, the control torque τ_t for a 1D joint degree-of-freedom (DOF) is determined by linear control

$$\tau_t = k_p(\hat{q}_t - q_t) - k_d\dot{q}_t, \tag{3}$$

where q_t and $\hat{q}_t$ are the current and reference values of the joint angle.

The complete optimization for a given window is then

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{q}_{1:T}). \tag{4}$$

For brevity, we drop the dependence of the objective on the motion in the remainder of the paper. Key-poses are parametrized by Euler angles for spherical joints.

We divide the optimization problem into spacetime windows [3], [39], because directly optimizing a long-duration motion would be prohibitively expensive. We find that windows are a convenient way to specify complex motions. Windows can be used for specific phases of gait or

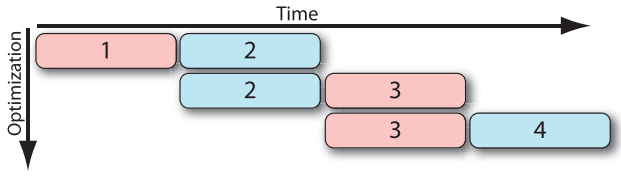


Fig. 1. Spacetime window optimization schedule. We define the objective function for a motion by creating a sequence of spacetime windows. Each window defines the objective function for a fixed interval. Optimization is performed with an advancing schedule: the motion during windows 1 and 2 are optimized. Then the motion in windows 2 and 3 are jointly optimized, and so on, until every window has been visited.

the relevant phases of other motions. Each window represents a fixed duration of the motion; we use $0.5\,\mathrm{s}$ windows. We do not consider problems where a long-term look-ahead is required; hence, we employ a simple, advancing optimization schedule (see Fig. 1). First, the first two windows of the motion are optimized together. Then, results for the second window are discarded and a new optimization is performed over the second and third window, holding the first window fixed. This process advances along pairs of windows until the entire motion is complete. This pairwise schedule allows each window to take the goals of the next window into account. As we discuss later, although we use fixed duration windows this approach enforces only a loose constraint on motion timing. Multiple passes [3], [39] could be used for problems requiring more look-ahead.

We use covariance matrix adaption (CMA) [40] to perform optimization, initializing all problems with a zero vector, which corresponds to an upright standing posture (i.e., the static pose $\dot{\mathbf{q}}_{1:T} = \mathbf{0}$). Hence, unlike previous work, no user effort or motion data is required for initialization. We run CMA with a maximum of 18,000 generations and a population size of 16. The optimization is performed in parallel on a dual quad-core Intel E5355 CPUs and takes roughly 20 minutes per window. We use OpenMP to parallelize CMA across all cores. Examples described in this paper took between 1 and 15 hours, depending on the number of windows used. Once a window has been optimized, we reset the subsequent window and repeat the above described process. For the motions described in this paper, all generations of CMA are typically required for to converge on a solution.

We select servo gains k_p, k_d by hand and use the same values for all motions described in this paper; consult the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.325, for PD gains values and simulator details. For a $0.5\,\mathrm{s}$ duration window, we place a spline knot every $0.1\,\mathrm{s}$. The optimization limits the range of reference trajectories for all joints (see Table 1 in the online supplemental material). Torque limits are enforced by clamping joint torques in the simulation; flips use torque limits of $\pm 400\,\mathrm{Nm}$, all other motions use $\pm 200\,\mathrm{Nm}$. The actual generated motion $\mathbf{q}$ has a joint limit only on the toe. Since we jointly optimize two windows at a time, and our character model has 35 actuated DOFs, the total number of variables for each pairwise optimization is 350. Motions where left-right body symmetry is enforced require 220 variables.

In the remainder of the paper, we describe objectives required to produce a variety of different motions. We first introduce objectives needed to generate variations of in-place balancing behaviors. More complex locomotion and acrobatic motions are described in subsequent sections. All motions can be seen in the accompanying video.

4 Balancing and Getting Up

We begin with a balance motion, which aims at placing the character in standing balance. We then show how it can be used to produce motions in which the character gets up from lying-down positions. While the objectives we use are quite simple, we believe it is surprising that these complex motions, which involve many contact changes, can be achieved from such simple specifications and optimizations.

4.1 Balancing

The standing balance goal entails optimizing the weighted sum of four objective terms over a given window. The first term

$$E_{COM} = (\mathbf{c} - \bar{\mathbf{c}})^2, \tag{5}$$

moves the COM $\mathbf{c}$ to a target position $\bar{\mathbf{c}}$ at a user-specified height over the approximate centroid of the base-of-support (we use the average positions of the centers of the feet projected on the plane). In our examples, the target height is $1.09\,\mathrm{m}$. The second term keeps the feet on the ground

$$E_{feet} = y^2_{leftFoot} + y^2_{rightFoot}. \tag{6}$$

The position of the feet on the plane is not specified. These first two terms are only active *at the end of the window*; that is, these terms aim to produce a motion that ends in a balanced state, without constraining how the character gets there. Note that the exact instant that standing balance is attained is not specified; the optimal motion could reach the final pose before the end of the window.

The third term penalizes control torque over the entire window

$$E_{torque} = \sum_{j,t} \tau^2_{j,t}, \tag{7}$$

where the summation is over all joint DOFs j and time indices t within the window. This term is used in all motions in this paper.

Finally, a rest pose term

$$E_{restPose} = \sum_{j \in \mathcal{J}} \sum_{t} \left(\theta_{j,t} - \bar{\theta}_{j,t}\right)^2, \tag{8}$$

is used to ensure that the character is standing upright, with arms in desirable positions. $\theta_{j,t}$ is the angle representation for joint j at time t and $\mathcal{J} = \{\text{lumbar, thorax, shoulder, elbow}\}$. We find that constraining this subset of joints, when combined with the other terms, is sufficient to achieve a reasonable rest pose. This term is only used when a balance window is used as the final window of the optimization.

4.2 Getting Up: Athletic

The balance objective in the previous section can be used directly for getting-up (see Fig. 2, top row), since it aims to

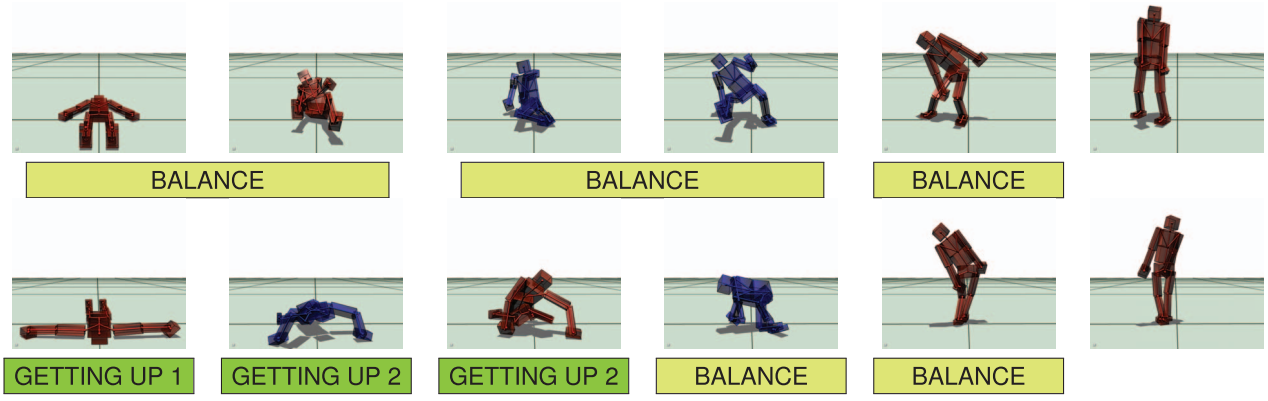


Fig. 2. Getting up motions. Top row: Starting from a supine position, optimizing over a sequence of four balancing windows yields a very athletic getting-up motion. The objective function is the same in each window, except for the rest pose term used in the final window. For each window except the last, the first and middle frames are shown. For the last window, the first and last frames are shown. Bottom row: A more normal getting-up motion from a prone position is produced by partially specifying contacts, for various windows, as described in the text. Each image shows the first frame for the corresponding window, except for the last image that shows the final frame.

end the window in standing balance. Despite the lack of explicit contact planning, the optimization successfully finds viable contact sequences needed to produce standing-up motions. In the accompanying video, we show examples of getting up from prone and supine positions. The resulting motion is very quick and exhibits highly athletic abilities.

4.3 Getting Up: Low-Energy

We can produce more typical getting-up motions by performing the optimization in a sequence of four windows, and by partially specifying the contacts for each window (see Fig. 2, bottom row). All windows include the torque-squared efficiency term E_{torque} (7) and use variants of the balance objective. In the first window, we keep the character prone, while raising his body. To accomplish this, three objectives are used: torque minimization E_{torque}, contact constraints, and target COM height:

$$E_{ground} = \sum_t \left(y_{lhand}^2 + y_{rhand}^2 + y_{ltoe}^2 + y_{rtoe}^2 + y_{lknee}^2 + y_{rknee}^2\right), \quad (9)$$

$$E_{COMh} = \left(y_{COM} - \bar{y}_{COM}\right)^2. \quad (10)$$

The E_{ground} objective penalizes nonzero height for each corresponding point on the body during the entire window. The E_{COMh} objective raises the COM to a target height $\bar{y}_{COM} = 0.3\ \mathrm{m}$ at the end of the window. In the second window, the character raises the right side of his body by pushing off the ground with his left limbs. This is done with the same objectives as above, but removing the left knee term, and increasing the COM target height to $0.5\ \mathrm{m}$. In the third window, we deactivate all but the left hand, left toe, and left knee constraints, and raise the COM target to $0.55\ \mathrm{m}$. The last window is a normal balance window.

We use a similar approach to generate a getting-up motion starting from a supine position. During the first window, objectives are used that encourage the character to get his back off the ground. We also place the hands and the feet on the ground during the entire window. During the second window, objectives keep the left hand on the ground and we include E_{COMh} with a COM target height of $0.45\ \mathrm{m}$. The last window uses the balance objective.

5 Walking

In this section, we show that full-body walking can be produced entirely through optimization with simple and human-interpretable objectives, and without relying on motion capture. We use one window for each swing phase (see Fig. 3), i.e., one period from toe-off to heel-strike. The same objective function is used for each window, with feet handling swing and stance tasks alternating from window to window.

In total, six objective terms are used in each window. These terms control the desired distance and direction of travel, heading, balance, foot contact, angular momentum, and energetic efficiency. Different parameters to the heading and distance/direction terms are used to produce forward walking, backward walking, and turns.

We specify a desired step-length and direction, by a vector $\mathbf{v}$. For example, if we wish the character to move $0.5\ \mathrm{m}$ along the x-direction in one swing phase, then $\mathbf{v} = [0.5, 0, 0]^T$. We evaluate this as

$$E_{stepdist} = \|(\mathbf{c}_E - \mathbf{c}_S) - \mathbf{v}\|^2, \quad (11)$$

where $\mathbf{c_S}$ and $\mathbf{c_E}$ denote the COM location at the start and end of the window.

The character's heading is controlled by

$$E_{heading} = (\alpha - \bar{\alpha})^2, \quad (12)$$

where α and $\bar{\alpha}$ are the actual and the desired character heading of the character's pelvis, at the end of the window.

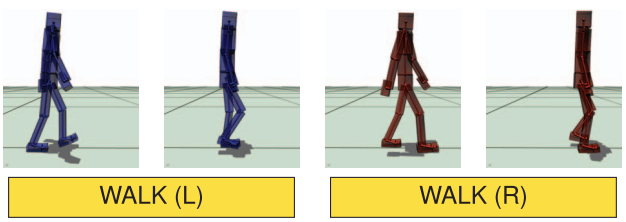


Fig. 3. Walking. First/middle frames of two successive walking windows. The objective function is the same in each window, except for swapping roles of stance/swing feet. Changing parameters of relevant objective function terms gives backward walking and turns, as shown in the accompanying video.

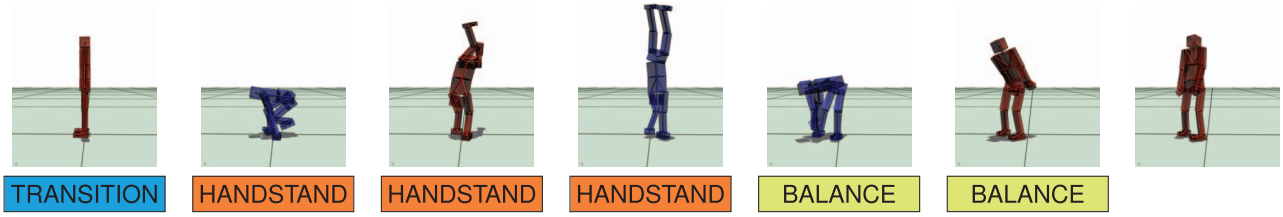


Fig. 4. Handstand. The character begins in standing balance. After a transition window, handstand windows (with the same objective function) are used to achieve the task. Finally, the character returns to standing using two balance windows.

We define heading as the right-handed angle about the axis perpendicular to the ground plane, measured from the x-axis.

For heel-strike at the end of the window, we use

$$E_{swingHeel} = y^2_{swingHeel}, \tag{13}$$

where $y_{swingHeel}$ is the position of the swing foot heel.

We want the stance foot to stay in contact with the ground, but we also want foot rolling, therefore we penalize vertical displacement of the stance toe $y_{stanceToe}$ from the ground over the whole window

$$E_{stanceToe} = \sum_t y^2_{stanceToe}. \tag{14}$$

At heel-strike, the position of the COM projected on the plane is roughly at the center of the base of support [41]. For this reason, we include E_{COM} (5), with a desired COM height that depends on the desired step length.

For walking motions, we modify the pose parameterization slightly, following Wang et al. [37]. Specifically the shoulder angle in the sagittal plane is coupled to the hip angle as

$$\theta_{lshoulder} = \omega(\theta_{rhip} - \theta_{lhip}), \tag{15}$$

$$\theta_{rhsoulder} = -\theta_{lshoulder}, \tag{16}$$

where ω is determined by the optimization. This reduces the number of free variables in the walking motion optimizations to 348 variables. Torques are also penalized using E_{torque}.

As demonstrated in the accompanying video, changing the heading direction and step distance can create forward walking, backward walking, and various sharp turns. Results are not entirely natural, in ways that reflect the biomechanical simplicity of the model.

6 Inverted Movements

In this section, we describe the objectives required to create a simple handstand and hand walk.

6.1 Handstand

A handstand window requires five energy terms that are applied over the entire window. Terms keep the hands on the ground and encourage the feet and the COM to be as high as possible

$$E_{hands} = \sum_t y^2_{lhand} + \sum_t y^2_{rhand}, \tag{17}$$

$$E_{feetV} = -\sum_t y_{lfoot} - \sum_t y_{rfoot}, \tag{18}$$

$$E_{COMV} = -\sum_t y_{COM}. \tag{19}$$

To penalize motions with head/ground contact, we use an objective

$$E_{headHeight} = \sum_t \delta(y_{head}, \zeta_{head}), \tag{20}$$

$$\delta(y, \zeta) = \begin{cases} C, & \text{if } y < \zeta, \\ 0, & \text{otherwise}, \end{cases} \tag{21}$$

that keeps the head height y_{head} above the threshold $\zeta_{head} = 0.35$ m for the entire window by applying a large penalty C to frames that violate the constraint. We use $C = 1{,}000$. The E_{torque} (7) term is also included.

In Fig. 4, we illustrate a case where the characters starts from standing balance, goes to the handstand, and returns to standing balance. The standing-balance-to-handstand transition window contains two terms, one to ensure that the COM stays above a height threshold $\zeta_{COM} = 0.6$ m:

$$E_{COMHeight} = \sum_t \delta(y_{COM}, \zeta_{COM}), \tag{22}$$

and the torque penalty E_{torque}. This transition window allows the character to enter a suitable position to begin the handstand motion; omitting it leads to a motion that achieves the handstand with superhuman speed and interpenetration. The handstand-to-standing-balance transition windows contain the same objectives as the balance motion with the addition of the $E_{COMHeight}$ term. We use a similar approach to create transitions with other motions.

The handstand motion is achieved in an aggressive manner: the character pivots entirely on his shoulders to achieve an inverted position, rather than taking a step, to build up momentum, as most humans would. An additional constraint for this step could be added if desired. The handstand itself is also more wobbly than a normal human handstand, which we suspect is due to the high precision or stiffness required for this motion. A longer look-ahead may also improve the style of this motion.

6.2 One-Handstand

We also demonstrate the ability to stand on one hand. This is achieved simply by removing the constraint on the right hand in E_{hands} (17). As shown in the accompanying video, the character achieves a one-handed handstand, raising the right hand in the air while remaining balanced.

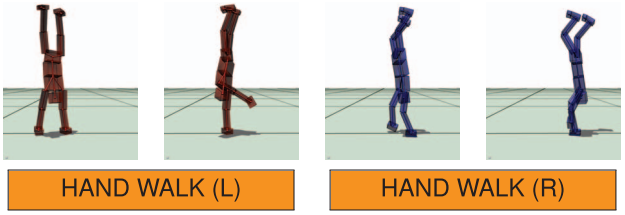


Fig. 5. Hand walk. This motion combines objective terms from the handstand and from regular walking. The first and middle frames of each window are shown.

6.3 Hand Walk

We can combine elements of the handstand and walking objectives to generate a motion where the character walks on his hands. To do this, we reuse the E_{hands} (17) but only consider a single contact hand in each window, alternating hands between windows. Including $E_{heading}$ (12) and $E_{stepdist}$ (11) terms introduced in the walking objective generates the hand walk (see Fig. 5). Stylistically, the hand walk and handstand are similar; transitions are somewhat athletic and the character is wobbly once inverted.

7 Rotational Movements

We now describe novel highly rotational movements. In each case, large rotations are achieved by using an objective which encourages high angular momentum about a given axis.

7.1 Headspin

The headspin motion is a difficult behavior seen in breakdancing routines. During this motion the character spins quickly on his head (see Fig. 6). The headspin employs the same objectives as the handstand, except that the $E_{headHeight}$ (20) term is modified to ensure that the head remains close to the ground (i.e., below 0.35 m).

In addition, we seek to maximize the angular momentum about the vertical axis during the entire window

$$E_{AM} = -\sum_t L_y^2. \tag{23}$$

We also penalize horizontal motion of the COM between the start and end of the window

$$E_{horiz} = \|\mathbf{c}_{horiz,E} - \mathbf{c}_{horiz,S}\|^2, \tag{24}$$

denoted with subscripts S and E, respectively.

In the accompanying video, we show a sequence where the character starts from standing balance, spins on his head, and returns to standing balance. Two windows are necessary for the character to transition from standing balance to the headspin. The first window only contains the E_{torque} (7) term. For the second window we add the E_{hands} (17) term. This has the effect of lowering the character's COM and encouraging hand placement on the ground in preparation for the headspin. To end the headspin motion, we use one window containing E_{torque} only. This results in the character lying on the ground. From that point on, balance windows (see Section 4) are used to transition to standing balance.

We note that the character transitions from standing balance to the headspin very rapidly. A breakdancer would need more time to carefully position himself to start the headspin. This can be taken into account by designing additional transition windows.

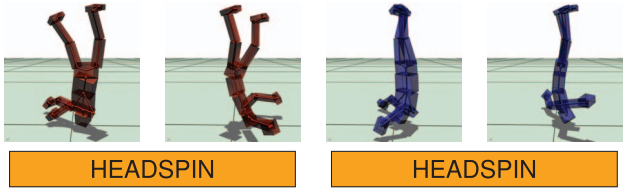


Fig. 6. Headspin. The character keeps his head on the ground while producing large angular momentum.

7.2 Handspin

The handspin motion is a breakdancing maneuver where the character spins on his hands (see Fig. 7). The handspin cost function consist of five terms, each described below.

As with the headspin, the handspin requires building up substantial whole-body angular momentum about the vertical axis. To accomplish this, we attempted to include objective terms that maximize angular momentum (e.g., (23)), however this generated inhumanly fast spinning behaviors. Instead, we found that using the objective

$$E_{AMRange} = -\sum_t \phi(L_y, \zeta_l, \zeta_u), \tag{25}$$

$$\phi(L_y, \zeta_l, \zeta_u) = \begin{cases} L_y^2, & \text{if } \zeta_l < L_y < \zeta_u, \\ 0, & \text{otherwise}, \end{cases} \tag{26}$$

which seeks maximum thresholded cumulative angular momentum, over a window, produced more appealing results. In all our demos we use $\zeta_l = -10^9$ N m s and $\zeta_u = 0$ N m s.

To ensure the head does not touch the ground during the handspin, we use $E_{headHeight}$ (20) with a 0.35 m threshold. We need the E_{hands} (17) term to force the hands on the ground. With the objective terms covered so far, we found that the body tended to be too close to the ground for a typical handspin. We include E_{COMV} (19) to address this. The E_{torque} (7) term is included.

Although most aspects of the motion are not explicitly planned, many strategies for generating angular momentum, matching those of breakdancers, emerge automatically. First, the character drops to the ground and spreads his legs. Next, legs are moved in opposite directions; the top leg is quickly moved forward while the bottom leg simultaneously pushes backwards off the ground. Once spinning, the character uses his feet to push off the ground at intermittent intervals to continue rotating.

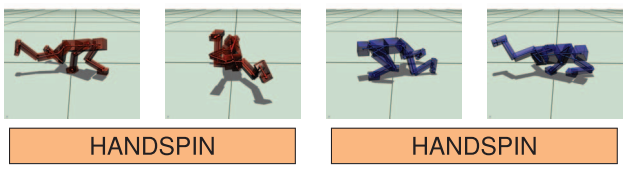


Fig. 7. Handspin. The character keeps his hands on the ground while producing large angular momentum.

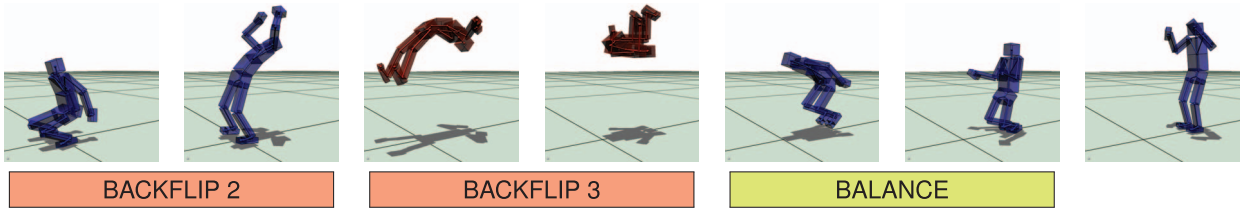


Fig. 8. The backflip is produced by controlling COM height, contacts, and maximizing angular momentum. The first and middle frames of windows 2 to 4 are shown, together with the final frame.

7.3 Flips

The flip motions consist of a rotation in the sagittal plane while the character is airborne (see Fig. 8). This is the first time that flips are generated without prior data; indeed, de Lasa et al. [29] report being unable to identify suitable features for creating flip controllers.

We divide the motion into four windows and enforce left-right body symmetry in the reference trajectory. All the windows include $E_{headHeight}$ (20) and $E_{COMHeight}$ (22) to prevent the head and the COM from getting too close to the ground (i.e., less than 0.6 m). The E_{torque} (7) term is also included. No additional terms are included in the first and third windows.

During the second window, we use E_{COMh} (10) with a target COM height between 1.4 and 1.6 m, depending on the desired style of the flip.

To generate a backwards flip, we include a term that maximizes angular momentum about the COM in the sagittal axis $\mathbf{d}$ during the entire window

$$E_{AMRange} = -\sum_t \phi(L_{\mathbf{d}}, -\infty, 0). \tag{27}$$

The only difference between the backwards and forwards flip is the sign of the angular momentum term $L_{\mathbf{d}}$.

This is a case where we found it difficult to resolve different objectives "fighting" each other. Specifically, if a large weight is used for $E_{AMRange}$, the character rotates but does not become airborne; however, if a large weight is used for E_{COMh} the character does not rotate sufficiently. Following [37], we use a thresholded quadratic term

$$E_{COMhQ} = Q(y_{COM} - \bar{y}_{COM}, \epsilon), \tag{28}$$

$$Q(d, \epsilon) = \begin{cases} d^2, & \text{if } |d| > \epsilon, \\ 0, & \text{otherwise}, \end{cases} \tag{29}$$

with $\epsilon = 0.05$ m and set the weight on (28) to be very large (see Table 2 in the online supplemental material). This objective penalizes COM displacement over the threshold and provides no penalty otherwise.

The fourth window uses the balancing objectives (see Section 4.1).

The accompanying video shows several flip variations. When flipping backwards arms swing forward quickly, during decompression, to generate needed angular momentum prior to ballistic motion. Near the apex of the aerial phase, the character tucks rapidly, to increase angular velocity, before extending his legs for landing. Once he lands, arms counter-rotate to aid balance. To generate needed rotations for the forward flip, the character takes an in-place hop and enters a tuck. The tuck (which resembles a sitting position) is held for nearly the entire movement. Hence, the character lands with bent knees and his hands can be seen touching the ground. A tighter tuck might help avoid high-loads on joints exhibited in the current motion. Alternatively, the forward flip could be initiated once the character has a greater forward velocity.

8 Ground Movements

This section describes two novel motions that keep hands and feet on the ground: push-ups and crawling.

8.1 Push-Ups

Push-ups use four objectives. The first two encourage hand and toe links to remain on the ground. For the hands we use (17) while

$$E_{toe} = \sum_t y_{ltoe}^2 + \sum_t y_{rtoe}^2, \tag{30}$$

is used for the toes. To lower and raise the character's body, we use

$$E_{COMh} = (y_{COM} - \bar{y}_{COM})^2, \tag{31}$$

and vary the COM target height between windows. In our examples, $\bar{y}_{COM}$ is set to 0.25 and 0.45 m. The E_{torque} (7) term is also used.

8.2 Crawling

Crawling is a quadrupedal gait that uses both arms and legs for locomotion (see Fig. 9). We divide crawling into two phases. In each phase, diagonally opposite limbs are considered to be in either stance or swing, with roles alternating from phase-to-phase. We optimize one window for each phase and use eight objectives for each window.

To encourage stance limbs to remain firmly planted we use two sets of objectives. Stance limb motion parallel to the ground plane is penalized by objectives

$$E_{stanceHand} = \|\mathbf{p}_{hand,E} - \mathbf{p}_{hand,S}\|^2, \tag{32}$$

$$E_{stanceKnee} = \|\mathbf{p}_{knee,E} - \mathbf{p}_{knee,S}\|^2, \tag{33}$$

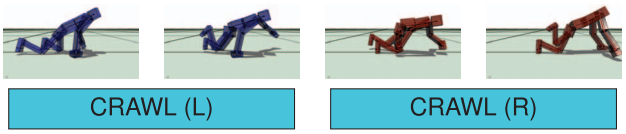


Fig. 9. First/middle frames of successive crawling windows.

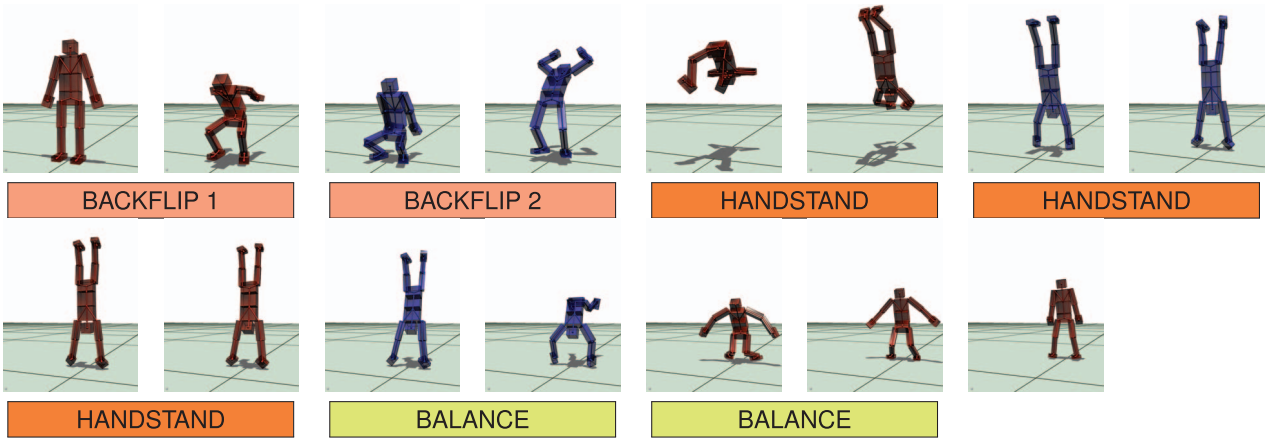


Fig. 10. Backflip-to-handstand. The first and middle frames of each window in a backflip followed by handstand and balance. The last figure is the final frame.

$$E_{stanceToe} = \|\mathbf{p}_{toe,E} - \mathbf{p}_{toe,S}\|^2, \quad (34)$$

where $\mathbf{p}$ denotes the position of each stance limb (i.e., hand, knee, toe), projected onto the ground plane, at the start S and end E of each window. Stance limb motion perpendicular to the ground plane is penalized using E_{ground} (see (9)). We reuse previously described objectives for other aspects of the motion: $E_{headHeight}$ (20) and $E_{COMHeight}$ (22) ensure the head and the COM are at least $0.3\,\mathrm{m}$ above the ground, while $E_{stepdist}$ (11) encourages movement in the desired direction. The E_{torque} (7) term is also included.

9 Longer Motions

It is straightforward to generate longer motions by concatenating different spacetime windows. Previous sections have presented several combinations of getting-up with acrobatic moves. Here we describe several longer sequences obtained by cocatenating previously described windows. Concatenation is simplest when one window ends in a good initial pose for the next window; for example, the character can transition directly from a handstand window to a hand walk window. Other combinations require transition windows (see Section 6.1).

9.1 Backflip-to-Handstand

Fig. 10 demonstrates a backflip followed by a handstand, followed by standing balance. No transition windows were required to generate this sequence. It is interesting to note that the character can enter the handstand phase even if he is entirely airborne.

9.2 Multiple Backflips

Fig. 11 demonstrates a challenging sequence of two backflips. The transition from the first to the second backflip is achieved by the balance objectives, with a COM target height of $0.8\,\mathrm{m}$ for E_{COM} to ensure that the character is slightly crouched. At this point, the character is in a good position to jump, hence we proceed with window 2 of the backflip motion. A minor modification is required to generate a flip motion where the character uses his hands to rotate. Specifically, the $E_{headHeight}$ term is modified to allow the head to get as close to $0.35\,\mathrm{m}$ from the ground.

9.3 Long Sequence of Moves

In the accompanying video, we demonstrate several long motions sequences that combine many previously described actions.

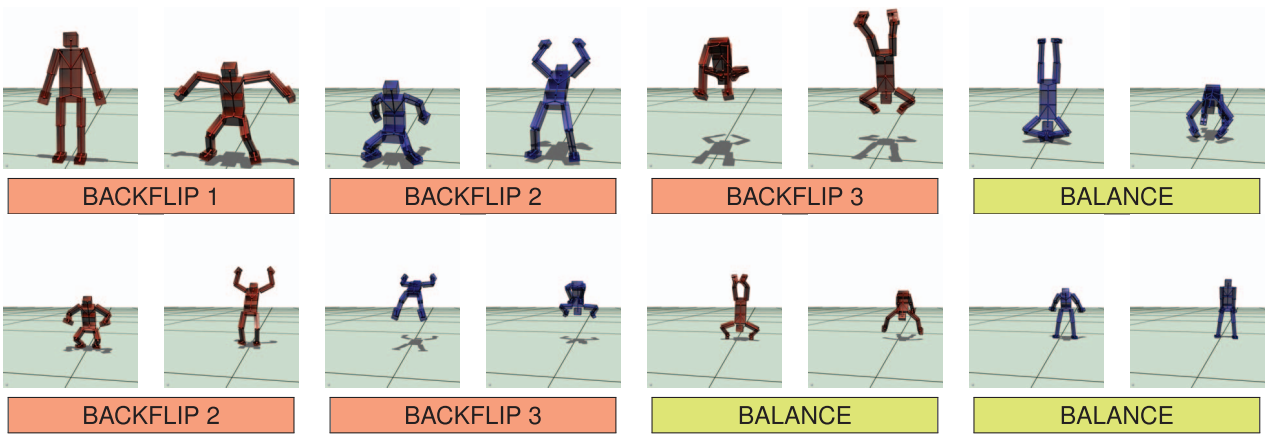


Fig. 11. Multiple backflips. A sequence of backflips is generated by combining windows from the backflip and balance motions. The first and middle frames of each window of the sequence are shown. The last image is the final frame.

10 Discussion

We have presented the first method for full-body trajectory optimization without relying on motion capture, specified key-poses, or periodicity. We show that a small set of simple objectives is sufficient to synthesize a wide range of movements. Some of the most complex movements, such as breakdancing maneuvers and flips, have not previously been generated by any other physics-based method.

An important characteristic of our method is that it does not require contacts to be predefined; contacts that are specified are of the form "this end-effector should touch the ground somewhere at the end of this window." Additionally, our method does not require the user to specify complete full-body pose at regular intervals. Combined, these factors makes it easier to generate a large variety of behaviors.

Overall, we did not find tuning the parameters to be particularly time consuming. As can be seen in the online supplemental material, most of the weights are powers of 10, and many of the objectives have the same weight across different motions. We believe it should be straightforward to apply our approach to other problems, such as for other types of animals [28], [16], and other types of terrain [28], [32], [33]. To create new motions, we first define the approximate sequence of required windows and objectives, starting from the end state of an existing behavior, and optimize each window in sequence. We author each window in sequence, caching results for previous windows, avoiding costly recomputation. As we gained experienced designing motions using this approach, we found that we could reuse the small set of objectives we described in the paper for a wide range of motions.

One limitation of our work is that the optimization is over a short time horizon. Some motions, like running to make a long jump over an obstacle, require longer term planning. Future work could involve combining our approach with long-term planning so that the character can autonomously navigate complex environments. Another possibility is to design an interactive system where an animator can build long sequences of motions by combining individual motions as building blocks [3]. The animator could have control over the timing of the spacetime windows and the parameters of a particular motion (e.g., the height of a backflip).

In most cases, we have not achieved perfectly natural human movement. For example, the synthesized walking motion includes some awkward hip movement. This can be avoided by restricting the hip joint limits in the reference trajectories (Table 1 in the online supplemental material) for walking, but not for the other motions that need a larger range of movement (e.g., flips). Such hand-tuning may no longer be necessary if we use biologically inspired objectives and musculotendon models [1]. We suspect that doing so can greatly improve the style of the motions in our work, as we currently use a very rough approximation of physical energy as an objective (7). Enabling interlimb contacts in our simulator would also remove interpenetration artifacts from results.

At present, it remains unclear what is the best way to stylize physics-based animation. Previous methods, such as specifying specific poses (e.g., [27]), or learning styles from examples [13] should be applicable in our approach as well. Another possibility is to include additional objectives that seek to replicate reference recordings (e.g., [15]).

Our approach shows that full-body trajectory optimization is feasible, but currently quite slow. We did not at any point attempt to make the optimization faster. An exciting open problem is how to speed-up the trajectory optimization so that it can be used in a real-time setting, either in model predictive control or in combination with a controller. For example, our method could be optimized by the concurrently developed method of Tassa et al. [42]. This would be an important step toward the synthesis of complex movements in games and humanoid robots.

Acknowledgments

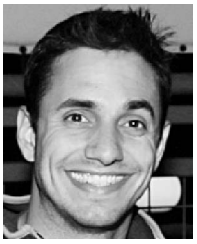
The authors would like to thank John Hancock for technical help. This work was supported in part by NSERC and CIFAR.

References

[1] J.M. Wang, S.R. Hamner, S.L. Delp, and V. Koltun, "Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives," *ACM Trans. Graphics,* vol. 31, 2012.
[2] A. Witkin and M. Kass, "Spacetime Constraints," *Proc. ACM SIGGRAPH,* pp. 159-168, 1988.
[3] M.F. Cohen, "Interactive Spacetime Control for Animation," *Proc. ACM SIGGRAPH,* pp. 293-302, 1992.
[4] Z. Liu, S.J. Gortler, and M.F. Cohen, "Hierarchical Spacetime Control," *Proc. ACM SIGGRAPH,* pp. 35-42, 1994.
[5] C.K. Liu and Z. Popović, "Synthesis of Complex Dynamic Character Motion from Simple Animations," *Proc. ACM SIGGRAPH,* pp. 408-416, 2002.
[6] A.C. Fang and N.S. Pollard, "Efficient Synthesis of Physically Valid Human Motion," *ACM Trans. Graphics,* vol. 22, no. 3, p. 417, 2003.
[7] S. Jain, Y. Ye, and C.K. Liu, "Optimization-Based Interactive Motion Synthesis," *ACM Trans. Graphics,* vol. 28, no. 1, pp. 1-10, 2009.
[8] M. Gleicher, "Retargetting Motion to New Characters," *Proc. ACM SIGGRAPH,* pp. 33-42, 1998.
[9] M. Gleicher and P. Litwinowicz, "Constraint-Based Motion Adaptation," *The J. Visualization and Computer Animation,* vol. 9, p. 65, 1998.
[10] Z. Popović and A. Witkin, "Physically Based Motion Transformation," *Proc. ACM SIGGRAPH,* pp. 11-20, 1999.
[11] Y. Abe, C.K. Liu, and Z. Popović, "Momentum-Based Parameterization of Dynamic Character Motion," *Proc. ACM SIGGRAPH/ Eurographics Symp. Computer Animation (SCA),* pp. 195-204, 2004.
[12] A. Safonova, J.K. Hodgins, and N.S. Pollard, "Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces," *ACM Trans. Graphics,* vol. 23, no. 3, p. 514, 2004.
[13] C.K. Liu, A. Hertzmann, and Z. Popović, "Learning Physics-Based Motion Style with Nonlinear Inverse Optimization," *ACM Trans. Graphics,* vol. 24, no. 3, p. 1071, 2005.
[14] K.W. Sok, M. Kim, and J. Lee, "Simulating Biped Behaviors from Human Motion Data," *ACM Trans. Graphics,* vol. 26, no. 3, p. 107, 2007.
[15] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu, "Sampling-Based Contact-Rich Motion Control," *ACM Trans. Graphics,* vol. 29, no. 4, Article 128, 2010.
[16] K. Wampler and Z. Popović, "Optimal Gait and form for Animal Locomotion," *ACM Trans. Graphics,* vol. 28, no. 3, p. 60, 2009.
[17] R.F. Nunes, J.B. Cavalcante-Neto, C.A. Vidal, P. Kry, and V. Zordan, "Using Natural Vibrations to Guide Control for Locomotion," *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games (I3D),* 2012.
[18] J.-C. Wu and Z. Popović, "Realistic Modeling of Bird Flight Animations," *ACM Trans. Graphics,* vol. 22, no. 3, p. 888, 2003.
[19] J. Tan, Y. Gu, G. Turk, and C.K. Liu, "Articulated Swimming Creatures," *ACM Trans. Graphics,* vol. 30, no. 4, p. 58, 2011.

[20] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of Complex Behaviors through Contact-Invariant Optimization," *ACM Trans. Graphics,* vol. 31, no. 4, p. 43, 2012.
[21] J.T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM, 2001.
[22] J.K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F. O'Brien, "Animating Human Athletics," *Proc. ACM SIGGRAPH,* pp. 71-78, 1995.
[23] M.H. Raibert and J.K. Hodgins, "Animation of Dynamic Legged Locomotion," *Proc. ACM SIGGRAPH,* pp. 349-358, 1991.
[24] W.L. Wooten and J.K. Hodgins, "Simulating Leaping, Tumbling, Landing, and Balancing Humans," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA),* pp. 656-662, 2000.
[25] K. Yin, K. Loken, and M. van de Panne, "SIMBICON: Simple Biped Locomotion Control," *ACM Trans. Graphics,* vol. 26, no. 3, p. 81, 2007.
[26] Y. Abe, M. da Silva, and J. Popović, "Multiobjective Control with Frictional Contacts," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA),* pp. 249-258, 2007.
[27] S. Coros, P. Beaudoin, and M. van de Panne, "Generalized Biped Walking Control," *ACM Trans. Graphics,* vol. 29, no. 4, p. 130, 2010.
[28] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion Skills for Simulated Quadrupeds," *ACM Trans. Graphics,* vol. 30, no. 4, p. 59, 2011.
[29] M. de Lasa, I. Mordatch, and A. Hertzmann, "Feature-Based Locomotion Controllers," *ACM Trans. Graphics,* vol. 29, no. 4, p. 131, 2010.
[30] A. Macchietto, V. Zordan, and C. Shelton, "Momentum Control for Balance," *ACM Trans. Graphics,* vol. 28, no. 3, p. 80, 2009.
[31] Y. Ye and C.K. Liu, "Optimal Feedback Control for Character Animation Using an Abstract Model," *ACM Trans. Graphics,* vol. 29, no. 3, p. 8, 2010.
[32] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust Physics-Based Locomotion Using Low-Dimensional Planning," *ACM Trans. Graphics,* vol. 29, no. 4, p. 71, 2010.
[33] J.-C. Wu and Z. Popović, "Terrain-Adaptive Bipedal Locomotion Control," *ACM Trans. Graphics,* vol. 29, no. 4, p. 72, 2010.
[34] M. da Silva, A. Yeuhi, and J. Popović, "Simulation of Human Motion Data Using Short-Horizon Model-Predictive Control," *Computer Graphics Forum,* vol. 27, no. 2, p. 371, 2008.
[35] M. da Silva, Y. Abe, and J. Popović, "Interactive Simulation of Stylized Human Locomotion," *ACM Trans. Graphics,* vol. 27, no. 3, p. 82, 2008.
[36] U. Muico, Y. Lee, J. Popović, and Z. Popović, "Contact-Aware Nonlinear Control of Dynamic Characters," *ACM Trans. Graphics,* vol. 28, no. 3, p. 81, 2009.
[37] J. Wang, D. Fleet, and A. Hertzmann, "Optimizing Walking Controllers," *ACM Trans. Graphics,* vol. 28, no. 5, p. 168, 2009.
[38] K. Yin, S. Coros, P. Beaudoin, and M. van de Panne, "Continuation Methods for Adapting Simulated Skills," *ACM Trans. Graphics,* vol. 27, no. 3, p. 81, 2008.
[39] C.K. Liu, A. Hertzmann, and Z. Popović, "Composition of Complex Optimal Multi-Character Motions," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA),* pp. 215-222, 2006.
[40] N. Hansen, "The CMA Evolution Strategy: A Comparing Review," *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms,* vol. 102, pp. 75-102, 2006.
[41] H. Herr and M. Popovic, "Angular Momentum in Human Walking," *J. Experimental Biology,* vol. 211, no. 4, p. 467, 2008.
[42] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS),* 2012.

Mazen Al Borno received the BSE degree in software engineering and the MS degree in computer science from McGill University, in 2008 and 2010, respectively. He is currently working toward the PhD degree in computer science at the University of Toronto, ON, Canada. His current research interests include computer graphics, character animation, scientific computing and optimization.



Martin de Lasa received the BESc degree in mechanical engineering and the BSc degree in computer science from the University of Western Ontario in 1998, the MESc degree in electrical engineering from McGill University in 2000, and the PhD degree in computer science from the University of Toronto in 2011. He is a technical lead and principal engineer at Autodesk where he leads animation activities. Prior to his doctoral studies, he worked at Boston Dynamics where he played a leading role in human simulation and robotics projects, including: digital biomechanics, BigDog, and LittleDog. His research interests include the areas of computer graphics and robotics, focusing on leveraging optimal control, optimization, and machine learning methods to build physically based models of motion to ease animation/control of simulated characters and robotic systems. He is the recipient of the CAIAC doctoral dissertation award for the top Canadian dissertation in artificial intelligence (2011).

Aaron Hertzmann received the BA degree in computer science and art and art history from Rice University in 1996, and the MS and PhD degrees from New York University in 2001. He is a professor of computer science at the University of Toronto, ON, Canada. He has worked at Pixar Animation Studios, University of Washington, Microsoft Research, Mitsubishi Electric Research Lab, Interval Research Corporation and NEC Research Institute. His awards include the MIT TR100 (2004), an Ontario Early Researcher Award (2005), a Sloan Foundation Fellowship (2006), a Microsoft New Faculty Fellowship (2006), a UofT CS teaching award (2008), the CACS/AIC Outstanding Young CS Researcher Award (2010), and the Steacie Prize for Natural Sciences (2010). He is a senior member of the IEEE.