To appear in ACM Transactions on Graphics 27(3)

Interactive Simulation of Stylized Human Locomotion

Marco da Silva Yeuhi Abe Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

Animating natural human motion in dynamic environments is difficult because of complex geometric and physical interactions. Simulation provides an automatic solution to parts of this problem, but it needs control systems to produce lifelike motions. This paper describes the systematic computation of controllers that can reproduce a range of locomotion styles in interactive simulations. Given a reference motion that describes the desired style, a derived control system can reproduce that style in simulation and in new environments. Because it produces high-quality motions that are both geometrically and physically consistent with simulated surroundings, interactive animation systems could begin to use this approach along with more established kinematic methods.

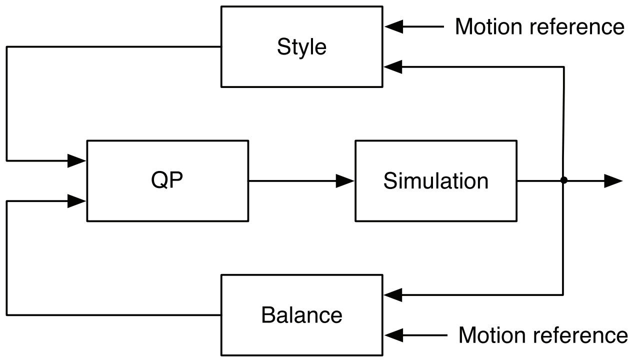

Figure 1: *Automated analysis of linear time-varying systems leads to a balance feedback policy that accounts for the underactuated dynamics in the human motion. With balance taken care of, a control system can then reproduce a given locomotion style by tracking every body joint. Quadratic programming combines the two feedback terms to compute control forces needed to simulate stylized locomotion in new environments.*

1 Introduction

Consistent interaction between characters, objects, and the surroundings can be difficult to achieve in dynamic environments. Simulation produces valid motions automatically but simulating human motion requires a control system that computes internal muscle forces. Without such forces, a simulated human would simply fall to the ground. The human form and the unstable dynamics of its bipedal locomotion make it difficult to find muscle forces that maintain balance, particularly in complex environments with unexpected disturbances. Its use in computer animation is further complicated by the need to generate motions that are comparable to recorded motion data. This has been difficult to achieve and, except for rag-doll effects, human simulation is rarely used in interactive animation systems.

This paper describes controllers for interactive simulation of stylized human locomotion. This control design precomputes a balance strategy for the given style using automated analysis of linear time-varying approximations. By tailoring the balance strategy in this manner, a controller preserves the style better than a more cautious strategy. For example, a controller can restrict the center of mass to be directly above the feet at all times, but in doing so it can only accomplish slow robotic-like motions. In contrast, the controller in this paper can reproduce a variety of locomotion styles.

The automatic precomputation, which typically completes in less than two minutes, determines a linear time-varying state feedback that prescribes desired accelerations for the three largest body segments: the two legs and the torso. Simultaneously, the second feedback loop tracks individual joint angles to compute the accelerations needed to preserve the given style. As shown in Figure 1, a reference motion guides both the style and balance feedback. The style feedback aims to preserve the nuances of the motion, while the balance feedback seeks to adapt the motion of three balance-critical segments. The control algorithm computes a final set of forces by maintaining a desired tradeoff between the balance and style feedback.

Interactive animation systems could use this approach in addition to traditional kinematic solutions. Physically based simulations automatically produce motions that are geometrically and physically consistent. When paired with a control system that produces high-quality motion, these simulations can animate lifelike human motion in dynamic environments, which is difficult to accomplish with kinematics alone. Furthermore, this process transforms a single recorded motion, valid for one environment only, into a general purpose action that can be used in many other settings or even composed with other actions to create versatile characters in games and other interactive applications.

2 Related Work

The ability to remain balanced affects the styles of motion a character can achieve [Wieber 2002]. The earliest controllers for legged creatures remained balanced by never allowing the projected center of mass to leave the base of support and by moving slowly enough so that dynamics did not have an impact on control [McGhee 1983; Raibert 1986]. This static balancing approach restricts the style of output simulations to slow, often unnatural motions.

An important advance was the ability to control legged creatures with active balance which allowed the simulation of more dynamic motions such as running and hopping and more agile walks [Miura and Shimoyama 1984; Raibert 1986]. Later these approaches were extended to create animations of humans executing specific tasks [Raibert and Hodgins 1991; Hodgins et al. 1995; Wooten and Hodgins 2000]. These approaches simplified control design by decoupling the problem into different regulation tasks, which were often understood by analyzing simple dynamics approximations. The controller we describe also approximates the full dynamics to make it possible to compute optimal feedback for the approximation. However, rather than designing these policies by hand, it precomputes feedback gains automatically. It also incorporates motion-capture data to ease the generation of more natural motions.

An impressive array of motions can be simulated using manually designed feedback controllers. These controllers can be composed to perform multiple tasks [Faloutsos et al. 2001]. However, these control policies are task, model, and environment specific. Changes in task, character model, or simulation environment require careful redesign of a new feedback policy and there is a nonintuitive relationship between controller parameters and simulation output. Dynamic scaling rules and randomized search can help [Hodgins and Pollard 1997] but it remains difficult for an artist to achieve a desired motion style using manually designed feedback controllers. In our approach, much of the control policy is computed automatically by exploiting an explicit model of the character dynamics. In addition, we take advantage of the fact that we know the desired motion ahead of time by precomputing a balance policy tailored to that motion. This reduces the number of manually adjusted parameters and improves the quality of simulated motion.

Others also take the approach of tracking a reference motion to improve the style of simulated motions. Earlier approaches based on tracking either avoided the issue of balance [Zordan and Hodgins 1999; Yamane and Nakamura 2003; Yin et al. 2003] or were restricted to tracking standing motions [Zordan and Hodgins 2002; Abe et al. 2007]. Yin and colleagues introduced a very effective balancing mechanism for SIMple BIped COntrol (SIMBICON), which they also coupled with feedback error learning to track cyclic walking motions [Yin et al. 2007]. The same balancing mechanism can also be coupled with quadratic programming to track non-cyclic motions without relying on feedback error learning [da Silva et al. 2008]. This paper explores another form of balance through subtle body adjustments instead of swing-leg placement. It is more complex to implement, but reduces parameter tuning and generates motions that are more similar to the given reference, including possibly non-cyclic motions such as standing, stepping, and transitions between the two.

Optimal control has been shown to produce stylistic human motions in *offline* simulations [Witkin and Kass 1988; Popović and Witkin 1999; Fang and Pollard 2003; Sulejmanpasić and Popović 2005; Safonova et al. 2004; Liu et al. 2005]. It is conceivable that offline trajectory optimization could be used to precompute a sequence of control actions that reproduce a motion style. However, these computations are finicky and slow. In interactive systems, they must be recomputed frequently because they are easily invalidated with unexpected changes in the dynamic environment. In fact, even minor accumulation of integration errors will disrupt such a policy after only a few simulation steps [Laszlo et al. 1996]. Laszlo and van de Panne apply limit-cycle control to automate the derivation of closed-loop controls from a given reference motion [1996]. Approximating the step to step return map dynamics has the advantage of incorporating collision effects and time-varying dynamics into the feedback policy. The balance policy in this paper relies on optimization and time-varying feedback to better match the given reference motion.

Recently, researchers have been able to simulate balanced motions by precomputing closed-loop control policies with randomized search algorithms [Ngo and Marks 1993; Tedrake 2004; Sharon and van de Panne 2005; Sok et al. 2007]. However, it is not known how to extend these approaches to higher dimensional search spaces, an important limitation given the large number of degrees of freedom for a 3D human character. Instead of relying on a brute-force search, our precomputation approximates the dynamics of the character with a discrete-time linear time-varying system (LTV) by differentiating system dynamics along the reference motion. This allows us to automate our analysis and to precompute balance policies efficiently. LTV approximations have previously been exploited for simple walking systems using differential dynamic programming [Jacobson and Mayne 1970; Atkeson and Morimoto 2003; Tassa et al. 2008]. In this work, we show how to apply controls derived for simple walking systems to more realistic 3D characters and how to incorporate desired motions from motion data.

3 Balance

A balance strategy determines the style of motion that can be reproduced in simulation. Conservative strategies may maintain balance by restricting the center of mass to be above footprints at all times. This is a safe strategy but it produces slow robotic motions. A regular walk requires excursions beyond these regions of support and more stylized motions may require even more delicate balancing. Manual design of a balance strategy is possible but tedious and it requires some knowledge of the expected terrain. Our goal is to devise an inexpensive control approach that tailors itself to the provided reference motion automatically. In this section, we describe the precomputation of a balance policy for a given reference motion.

3.1 Balance Objective

Maintaining balance requires careful manipulation of all limbs to adjust the location of the overall center of mass. Control designers often make simplifying assumptions to make this design easier, its analysis simpler, and its use more broadly applicable [Raibert 1986; Laszlo et al. 1996]. Full and Koditschek argue that simple models play an important role in neuromechanical hypotheses of motion control [1999]. Others have used simple models to simplify high-dimensional nonconvex optimizations [Popović and Witkin 1999; Fang and Pollard 2003; Safonova et al. 2004]. We, too, will make similar approximations for the stepping motions we are interested in controlling. What makes our approach unique is that we make the entire analysis automatic and its application efficient by exploiting the information in provided reference motions.

We formulate balance control as an optimization that tracks the provided reference motion. Consider a dynamical system with n degrees of freedom and m actuators. A reference motion for that system is a sequence of states $\bar{x}_1, \ldots, \bar{x}_T$, where each state $\bar{x}_t \in \mathbb{R}^{2n}$ includes generalized positions and velocities. The current tracking error is the difference between the current and desired state $\Delta x = x - \bar{x}$. An optimal controller, then, finds the sequence of control actions, $u \in \mathbb{R}^m$, that minimizes the total tracking cost:

$$\arg\min_{u} \quad \Delta x_T^\top Q_T \Delta x_T + \sum_{t=1}^{T-1} \Delta x_t^\top Q_t \Delta x_t + u_t^\top R_t u_t \qquad (1a)$$

$$\text{subject to} \quad x_{t+1} = F(x_t, u_t). \qquad (1b)$$

The system function F expresses the discrete dynamics of the system. We use the discrete expression instead of the continuous form, $\dot{x}(t) = f(x(t), u(t))$ because numerical integrators fundamentally assume constant control force $u(s) = u_t$ over finite intervals $s \in [t, t + \delta t]$ of short duration:

$$F(x_t, u_t) = x_t + \int_t^{t+\delta t} f(x(s), u_t)\,\mathrm{d}s \qquad (2)$$

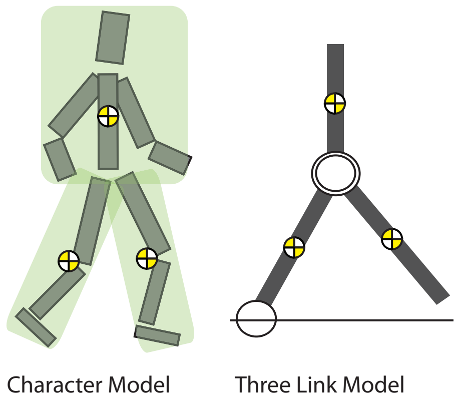

Figure 2: *A simple three link model is constructed from the geometric and inertial properties of the character model. The inertial properties of each simple link are created by summing the inertial properties of corresponding links on the simple character. The base link is attached to the ground with a three-dof ball joint. Two three-dof ball joints connect the other two links to the base link. The yellow dots represent centers of mass, which are in agreement with the detailed character model.*

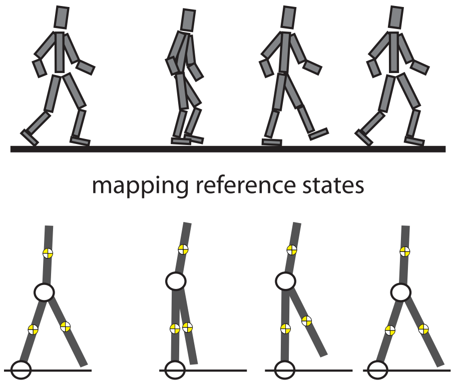

Figure 3: *The state of the character is geometrically mapped to the approximate model. The user annotates sections of the reference motion corresponding to different contact states. A simple model is constructed for each single support section of the reference motion and the motion is retargeted geometrically to produce a sequence of desired states for the simple model. At runtime, the same mapping is used to determine the current tracking error for the three-link model.*

The tracking and control costs are determined by the (semi) positive definite matrices, $Q_t \geq 0 \in \mathbb{R}^{2n \times 2n}$ and $R_t > 0 \in \mathbb{R}^{m \times m}$, that express the desired compromise or the optimal performance. For example, if Q weights the tracking error across all joints equally, then the optimal control will reduce tracking errors in the wrist motion as vigorously as it corrects errors in the upper leg motion. While this choice preserves the style of hand and leg motion alike, it can also lead to poorly balanced motion because the legs and the upper body play a more critical role in bipedal locomotion. Designing these tradeoffs is more difficult and less general when the model has many degrees of freedom. Instead, we use a simple three-link model (Figure 2) that suppresses many details while preserving only those aspects of dynamics that are most relevant for balance control.

The three-link model is underactuated like the full human character with its mass distributed over the three-links. These are important features for recreating human motion strategies which manipulate total linear and angular momentum to maintain balance [Abdallah 2005; Kajita et al. 2003; Pratt et al. 2006]. The base link approximates the contact between the foot and the ground with an unactuated ball joint. The next link is attached to the base with a ball joint like upper body in the human figure. This allows it to speed up and slow down by leaning forwards and backwards. Lastly, the third link is attached to the upper body with a ball joint just like a swing leg whose motion is used to anticipate the next contact. We then adapt the same control formulation in Equation (1) using the state of the simple model and the system function derived from the dynamics for the three-link model: $\dot{x}(t) = g(x(t), u(t))$, where $x \in \mathbb{R}^{18}$ consists of three rotations [1] and three angular velocities, and $u \in \mathbb{R}^6$. The inertial properties for the dynamics of the simple model are constructed automatically by accumulating inertial properties of corresponding links in the full model.

3.2 Computation of Balance Controls

Even with the simple model, interactive computation of the control policy according to Equation (1) remains impractical. It is a nonconvex optimization problem because of the nonlinear dynamics of the simple model. These problems can be solved with smooth optimization but they require a good initial guess and are still too slow for interactive simulation [Betts 2001]. Moreover, the solution would only be valid for a single initial condition. In interactive animation systems, we would also need to compute controls fast enough to reject any new disturbances.

[1] In practice, we represent rotations with quaternions making x slightly bigger but the same formulation works for euler angle representations.

Linear Time-Varying Approximation Our solution approximates the nonlinear dynamics of the simple model with a discrete-time linear time-varying system (LTV):

$$\Delta x_{t+1} = A_t \Delta x_t + B_t \Delta u_t. \tag{3}$$

The time-varying system matrices $A_t \in \mathbb{R}^{18 \times 18}$ and $B_t \in \mathbb{R}^{18 \times 6}$ are computed by linearizing the nonlinear system dynamics along the desired reference trajectories. Given the system function G for the simplified dynamics (cf. Equation (2)), the time-varying system matrices are the Jacobians $A_t = \mathrm{D}_1 G(\bar{x}_t, \bar{u}_t)$ and $B_t = \mathrm{D}_2 G(\bar{x}_t, \bar{u}_t)$ of G with respect to the first and second argument.

We calculate the reference trajectory, $\bar{x}$, using a simple geometric mapping from the character to the simple model (see Figure 3). The ankle of the supporting leg of the character is mapped to the base joint of the simple model. Then, the location of the center of mass of the support leg relative to the ankle is used to determine the angle of the base link on the simple model. This angle determines the location of the “hip” on the simple model. The relative locations of centers of mass of the upper body and swing leg are similarly used to determine the orientation of the remaining links on the simple model. The relative velocities of the centers of mass on the full character determine the angular velocities of the simple model.

We annotate sections of each reference motion to delineate different contact states. A simple model and associated LTV is constructed for each single support phase. In a simple walk cycle for example, there would be an LTV system for the right step and another for the left (Figure 4). The LTV systems do not span collision events.

We use iterative optimization to estimate the matching control forces $\bar{u}$ for the mapped reference motion [Li and Todorov 2004]. The optimization usually converges in less than five iterations and

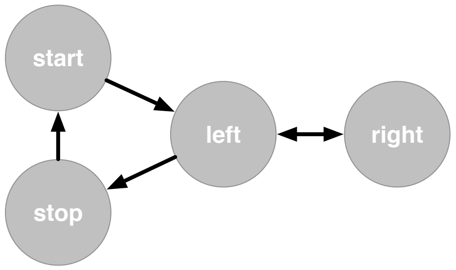

Figure 4: *The balance control approximates dynamics of each single support phase with a linear time-varying system. Dwell times are determined manually by timing corresponding phases in the reference motion. The balance control illustrated in this diagram alternates between LTV systems for left and right step with the option to pause and restart.*

the total time needed to compute the LTV matrices typically completes in less than two minutes. While it is convenient that these precomputations are quick, speed is not essential here because the LTV systems are constructed only once for each reference motion.

Feedback Policy The LTV approximation allows us to compute the optimal feedback policy for a problem with quadratic cost functions. We apply this idea to derive balance control from this approximation of the original formulation:

$$\min_{\Delta u} \quad ||\Delta x_T||^2 + \sum_{t=1}^{T-1} c||\Delta x_t||^2 + ||\Delta u_t||^2 \qquad (4)$$

$$\text{subject to} \quad \Delta x_{t+1} = A_t \Delta x_t + B_t \Delta u_t, \qquad (5)$$

The time window T is determined by the duration of the corresponding single support phase in the reference motion. The scaling constant c determines the tradeoff between tracking fidelity and additional control effort. We found a value of $c = 0.1$ yielded the best results in our experiments, but many other values will yield motion of similar quality.

The solution to this problem is a linear function of the tracking error, $\Delta u_t = K_t \Delta x_t$, which is known as the Linear Quadratic Regulator (LQR) [Stengel 1994]. The feedback gains K_t are precomputed efficiently by numerically integrating Riccati equations [Kalman 1960; Brotman and Netravali 1988; Stengel 1994]. This solution is optimal for the given LTV system. Differential dynamic programming is closely related to LQR and has been exploited for simple walking systems [Jacobson and Mayne 1970; Atkeson and Morimoto 2003; Tassa et al. 2008].

In addition to allowing us to formulate an effective balance-favoring objective, the simple model has three other benefits. First, it simplifies computation of LTV approximations. Multi-body simulators typically solve linear complementarity programs to compute frictional contact forces [Baraff 1994; Anitescu et al. 1998] which are non-smooth mappings of the current state. Instead, our simple model approximates this contact with an unactuated joint whose action is a smooth function of the state. Second, the computation of the optimal feedback gains is faster. The computation of LQR control is $O(Tn^3)$ where T is the number of time samples. Since the simple model has fewer degrees of freedom (9 versus 60), the computation of optimal feedback gains is faster. Third, the simple model reduces the memory needed to store the optimal feedback gains. The optimal gain matrix requires $O(Tn^2)$ space, which makes it less practical to store feedback gains for the full human model, especially for many reference motions. The simple model, on the other hand, requires only a small fraction of that space.

In principle, linear approximations could differentiate through contact transitions. For example, Popović and colleagues do so for rigid body simulations with frictionless impacts [2000]. We take a simpler approach by approximating each single support phase (e.g. alternating left and right steps for walking). During double support, the control relies on another feedback mechanism (§4.1) to stabilize the character. Delineating the reference motions in this way is needed given our particular choice of simple model. It simplifies contact dynamics and has the advantage of making it easier to chain different reference motions. A disadvantage of breaking the feedback policies apart is that they are no longer optimal for tracking an entire motion.

We have now described the precomputation portion of our controller which finds a set of time varying gain matrices to stabilize the motion of an approximation of our character. In the next section, we describe a didactic example that illustrates one application of our control: standing balance recovery.

3.3 Example: Balance Recovery

We illustrate the behavior of LQR control on a balance recovery task for our simple three-link model. Pratt and colleagues develop another balance strategy for an equivalent underactuated model [2006] and many other approaches are also discussed in the literature. Unlike most of this prior work, our approach relies on fully automated analysis with linear approximations and optimal state feedback. And we will use the same systematic approach to reduce the need for manual tuning when controlling more complex stylized motions.

The task for our control is to recover balance by returning to the "inverted-pendulum" configuration, an unstable equilibrium point for the model. The reference motion is a single pose indicating the inverted-pendulum configuration. The reference forces are zero in this case because the desired pose is a static equilibrium point. Since there is no particular time the pendulum should return to the equilibrium point, the optimal feedback forces are determined by the solution to an infinite-horizon problem:

$$\min_{\Delta u} \quad \sum_{t=1}^{\infty} \Delta x_t^\top Q \Delta x_t + \Delta u_t^\top R \Delta u_t \qquad (6)$$

$$\text{subject to} \quad \Delta x_{t+1} = A \Delta x_t + B \Delta u_t, \qquad (7)$$

where the discrete-time approximation is time-invariant because the linearization is around a single point instead of a trajectory.

The feedback control derived from this formulation is quite effective. For a model designed with rough human-like inertias and dimensions, the controller recovers for a range of leaning angles up to the maximum lean of 20 degrees. This maximum lean corresponds to a deviation of 36 centimeters between the projected center-of-mass and the base of the support leg.

The precise manner of recovery is controlled by the tracking Q and control R costs, with a very wide range of possible settings. We show one time-lapse sequence of recovery in Figure 5 for the case when the velocity errors are permitted by setting their costs to zero while both the position and control are weighed equally by setting their cost matrices to the identity. The controller recovers by rapidly swinging the upper body in the direction of the fall while simultaneously swinging the non-stance leg in the opposite direction. Once the center of mass is safely over the support leg, the model slowly returns to its balanced equilibrium.

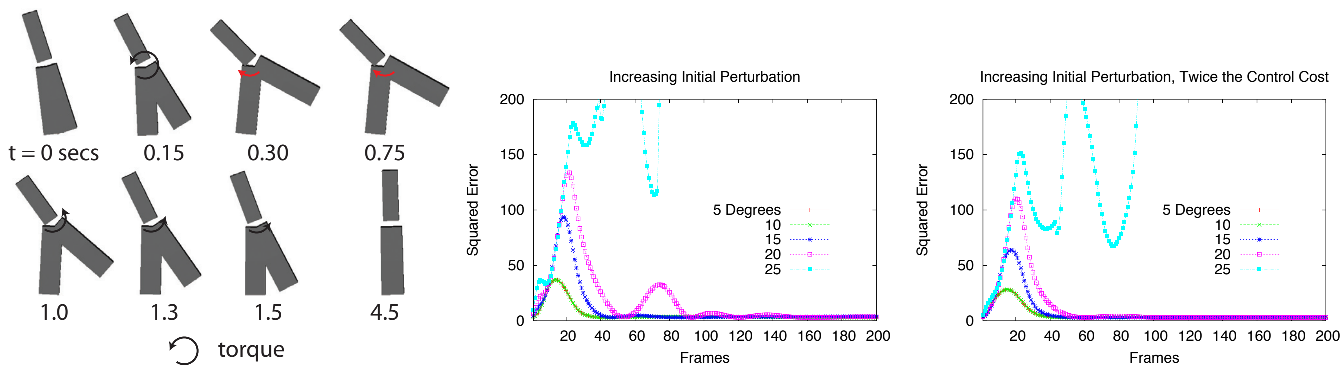

Figure 5: *To prevent a fall, the LQR balance controller employs a flywheel strategy, rapidly swinging the upper body in the direction of the fall and the swing leg in the opposite direction. The strategy is depicted using snapshots from a simulation in the leftmost image. The arrows indicate the direction and relative magnitude of the applied torque. For this particular model, the controller can recover from an initial error of about 20 degrees which corresponds to the projected center of mass being approximately 36 centimeters away from the base of support. The time evolution of the squared tracking error of this system is shown for various initial perturbations in the plots on the right. Doubling the control cost slightly improves performance but in either case the end result is the same.*

4 Balance and Style

The precomputed balance gains are tailored to a given reference motion in order to emulate the motions of two legs and the upper torso. The second control loop preserves finer nuances by tracking the entire state of the character. At runtime, the two loops, balance and style, produce different versions of the best action to take. We arbitrate between these two suggestions using a quadratic program that finds the feasible forces for a weighted combination of the two suggestions.

4.1 Style Feedback

The style feedback determines the joint accelerations a_s needed to track the reference motion:

$$a_\mathrm{s} = \ddot{\bar{q}} + k_1 D(\bar{q}, q) + k_2(\dot{\bar{q}} - \dot{q}), \tag{8}$$

where the comparison function D uses quaternion arithmetic to compute the angular acceleration needed to eliminate the difference between current and desired joint orientation [Barbič 2007]. This feedback acceleration tracks the root of the full human figure. So, although it primarily maintains style, it also plays a role in maintaining balance. In a standing motion, for example, any motion of the root produces accelerations that dampen this motion.

If desired joint accelerations are met with properly chosen control forces, the error on each joint will behave as a spring-damper mechanism. The stiffness k_1 and damping k_2 gains determine the rise, settling time, and the overshoot for the error-cancellation curve. We keep things simple in our experiments with gain settings for a critically damped spring $k_2 = 2\sqrt{k_\mathrm{s}}$, leaving only one parameter value k_s for each joint. The values change for each human figure, but in many cases, the same values are used for all motions with that character model (Table 1).

4.2 Balance Feedback

The balance feedback determines the stable motion of the simple model using the precomputed state feedback gains K_t. We compute this control by mapping the full state of the character $(q, \dot{q})$ onto the reduced state of the simple model x as described in Section 3.2. The balance control is then a function of measured deviations Δx and precomputed reference forces: $u_t = \bar{u}_t + K_t \Delta x_t$. To map this motion back onto the full character, we use forward dynamics of the simple model to compute the center-of-mass accelerations a_b for all three limbs in the simple model given the new u_t. The balance control for the full character will then seek to find joint accelerations that match center-of-mass accelerations of the simple model.

4.3 Quadratic Programming

Our control computes the final forces for the simulation with quadratic programming (QP). The QP solution arbitrates between style and balance feedback by finding joint torques that are close to a weighted combination of the two suggested accelerations. This is done carefully to ensure that computed torques are consistent with the feasible ground reaction forces:

$$\min_{u,\ddot{q},\lambda} \quad w_\mathrm{s}\|a_\mathrm{s} - \ddot{q}\|^2 + w_\mathrm{b}\|a_\mathrm{b} - \dot{J}_\mathrm{c}\dot{q} - J_\mathrm{c}\ddot{q}\|^2 \tag{9a}$$

$$\text{subject to} \quad \ddot{q} = f\left((q, \dot{q}), u + J_\mathrm{p}^\top V\lambda\right) \tag{9b}$$

$$J_\mathrm{p}\ddot{q} + \dot{J}_p\dot{q} = 0,\ \lambda_i \geq 0. \tag{9c}$$

The objective function is a weighted sum of the style and balance feedback terms. The balance feedback terms are desired accelerations for three points on the body and are active during single support. The Jacobian matrix J_c relates the acceleration of these points to the generalized configuration of the character. The constraints maintain a physically consistent relationship between joint accelerations $\ddot{q}$, control torques u, and contact forces $V\lambda$ by enforcing the contact dynamics equations at the current time instant [Abe et al. 2007]. Note that $u \in \mathbb{R}^{n-6}$ is the control signal for the character and not the simple balance model.

The manipulator equations (9b) encode the dynamics of the linked rigid body structure representing the character. These equations can be derived by Euler-Lagrange mechanics and can be evaluated efficiently [Featherstone and Orin 2000]. For a fixed position and velocity, the manipulator equations express a linear relationship between applied forces and acceleration. Similar expressions are exploited in computed torque methods to control industrial robots [Lewis et al. 1993]. Our formulation follows a more general formulation [Abe et al. 2007] that also includes unilateral constraints on ground reaction forces and mixed objectives.

The remaining constraint equations (9c) prevent the ground reaction forces from accelerating the contact points. The expression $J_\mathrm{p}^\top V\lambda$ maps external ground reaction forces expressed in a discretized friction cone basis, V, into generalized torques using the Jacobian for the contact points J_p [Fang and Pollard 2003]. The inequality constraint ensures that ground reaction forces do not pull on the character ($\lambda_i \geq 0$).

The behavior of the QP is controlled by the relative weights of the objective function, w_b, and w_s. In practice, it is not too difficult to find values of these parameters that work and similar values typically work for many motions on the same character. We use a recursive implementation of linked rigid body dynamics [Featherstone and Orin 2000] and SQOPT to solve QP problems [Gill et al. 1997].

Adjustment with PD. Following our formulation in previous work [da Silva et al. 2008], the controller adjusts the QP solution with a low gain proportional-derivative (PD) component. The desired pose (i.e. set point) of the PD component is the current pose of the reference motion. The gains for the PD component were set to gains used in previously published work [Sok et al. 2007; Yin et al. 2007] and then further reduced by multiplying with a factor smaller than one. Using lower gains allows the control to be less stiff which leads to more stable simulations. The values used are listed as the "Strength" parameters in Table 2. At run time, the gains for a joint are scaled by the composite rigid body inertia of all child links [Zordan and Hodgins 2002; da Silva et al. 2008].

The PD adjustments accomplishes two tasks. First, it guides the character through contact transitions. The QP's model of dynamics assumes a fixed contact configuration while we would like to track motions that make and break new contacts. Second, the PD component adjusts the QP solution at each simulation step. The QP step typically takes two to three milliseconds on a Pentium 4 2.8 Ghz processor. To allow the simulation to run faster, the QP is solved every ten to one hundred simulation steps. The PD component is calculated at each time step allowing it to react to disturbances instantly.

5 Results

The new controller produces high-quality motions for a large variety of reference motion styles. In many instances, simulations are almost indistinguishable from input motions. Moreover, it succeeds in the presence on unexpected disturbances, allowing it to generate physically valid motions even in the environments that are quite different from the one used during motion capture. Our comparisons with prior work reveal that customized balance strategy yields motions that better match the style of a given reference motion. The final results and these comparisons are best seen in our accompanying video.

5.1 Style

The automatic derivation of our controller simplifies simulation of stylized human motions. We demonstrate this versatility by simulating motions acquired from several different sources: the dataset assembled by Sok and his co-workers [2007]; motions generated by the SIMBICON controller [Yin et al. 2007]; and our own dataset of stylized motions recorded in a motion-capture studio. Time-lapse snapshots of some example simulations are shown in Figure 8.

Changing styles with our controller is straightforward. Once a good set of controller parameters have been found for one motion, those parameters typically work for many reference motions. Listed are the parameter settings for each of the styles simulated by our controller. Table 1 lists several of the styles simulated by our controller and the parameters that had to be changed. The first two datasets required very little tuning; the same parameter setting worked for all motions in many different environments. Simulating our own stylized motions required some tuning, but this still reduced to changing only two real-valued parameters.

Motion	k_s	w_b
2D Data		
Walk	500	0.1
Uphill	500	0.1
Backwards	500	0.1
March and Walk	500	0.1
3D SIMBICON		
Walk	1000	0.1
Downhill	1000	0.1
3D Mocap		
Sideways Steps	1200	0.2
Monty Python	1200	0.01
Monty Python See-Saw	1200	0.01
Stop and Start	1200	0.01
Dance	1800	0.01
Sumo Stomp	1200	0.01
Turn	1200	0.01

Table 1: *Changing styles with our controller is usually straightforward. Once a good set of controller parameters have been found for one motion, those parameters typically work for many reference motions. Listed are the parameter settings for each of the styles simulated by our controller. The style feedback parameter is k_s. Finally, w_b determines the weight of the balance feedback policy. The style weight, w_s, was fixed at 0.5 for every result.*

Highly stylized motions require more expressive simple models. For example, we were not able to track a motion where the subject swayed his arms and upper body from side to side to imitate a charging monkey. The simple model used in our experiments so far does not capture that balance strategy because it lacks arms, which play a critical role for the balance of this motion. We expect that adding additional links to our simple model would address this mismatch.

The balance policy used by a controller determines which types of motions can be simulated by that controller. In the extreme case, a controller that cannot balance will simply fall down which is probably not the style intended. The controller can be better at reproducing the reference style by customizing its balance policy for the given reference motion. In the video, we compare the tracking quality with two recently proposed controllers [da Silva et al. 2008; Sok et al. 2007]. Although some amount of tracking error is unavoidable (due to many differences between the simulation and the real world), the controller with the heuristic balance policy exhibits an irregular gait [da Silva et al. 2008]. In contrast, the controller with a tailored balance policy yields motions that are smoother and more faithful to the original reference motion. Direct visual comparison also shows reduced oscillation for a backwards walk when compared to the results presented by Sok and colleagues [2007].

5.2 Balance

An effective balance policy allows the controller to adapt the motion to new environments. In practice, the animator can change the terrain or the environment to produce numerous variations from a single reference motion. In the video, we adapt reference motions

2D Model					
Link	Strength	Mass	Inertia		
head	150	3	0.011		
upper arm	200	2	0.022		
lower arm	150	1	0.009		
torso	N/A	10	0.176		
thigh	200	7	0.121		
shin	200	5	0.077		
foot	200	4	0.019		
3D SIMBICON					
Link	Strength	Mass	Ix	Iy	Iz
hip	N/A	16.61	0.0996	0.1284	0.1882
trunk	30	29.27	0.498	0.285	0.568
head	1.5	6.89	0.0416	0.0309	0.0331
upper arm	1.5	2.79	0.00056	0.021	0.021
lower arm	1.5	1.21	0.00055	0.0076	0.0076
hand	0.1	0.55	0.05	0.2	0.16
thigh	9	8.35	0.145	0.0085	0.145
shin	3	4.16	0.069	0.0033	0.069
foot	0.3	1.34	0.0056	0.0056	0.00036
3D Mocap					
Link	Strength	Mass	Ix	Iy	Iz
hip	N/A	12.8	0.174	0.148	0.12
trunk	100,400	17.02	0.335	0.167	0.281
head	30,120	4.067	0.02	0.0176	0.0248
clavicle	40,100	2.51	0.0111	0.0111	0.00691
upper arm	40,100	1.42	0.0133	0.0014	0.0132
lower arm	30,120	0.57	0.003	0.0003	0.003
hand	30,120	0.09	0.00008	0.00002	0.0001
thigh	400,1600	9.013	0.204	0.0356	0.208
shin	400,1600	3.91	0.0585	0.0087	0.06
foot	100,400	0.29	0.0002	0.0004	0.0003
toe	5,20	0.23	0.0002	0.0002	0.0002

Table 2: *Model parameters used for our results. The joint stiffness values are low when compared to uses of these models in other controllers. This is because much of the control effort is generated by the QP component of our controller. The units are as follows: newtons per radian for the gains, kilograms for the mass, and kilogram meters cubed for the inertias. For the model used to track motion capture results, inertial properties were generated by integrating a fixed density over geometric models of each limb. The dimensions of the character were chosen to match the dimensions of our capture subject. Also, for this model, we used two sets of gains for the PD component of the controller as indicated by comma-separated strength values.*

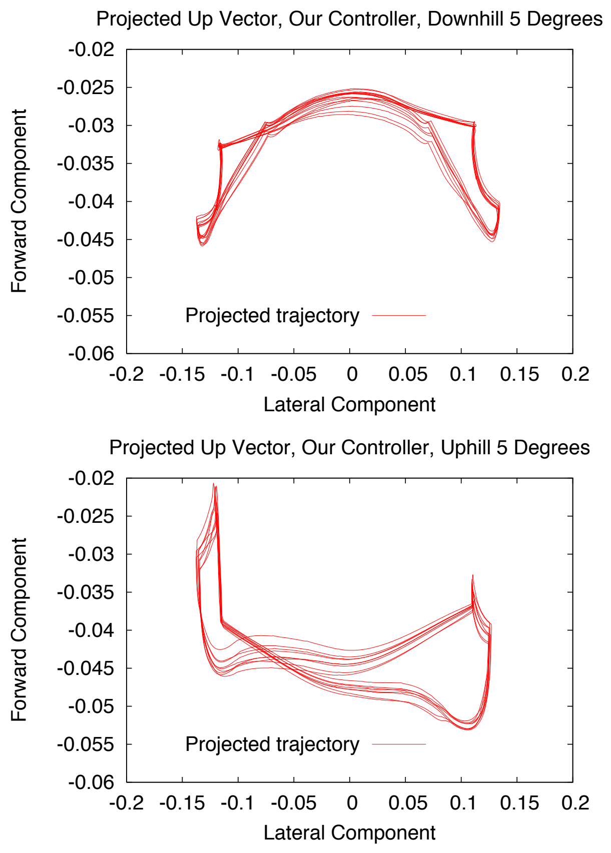

Figure 6: *We plot the horizontal components of the root link's up vector. The x axis is the lateral component and the z axis is the forward component (the character is walking down negative z). The first plot was created using our controller on a slope of five degrees tracking the output of a manually designed PD controller on flat ground. The second plot was created using our controller on an uphill slope of five degrees but with the same parameters as the downhill plot. Note that in the uphill plot, the up vector has a more negative forward component in the upright phase (lateral component equal to 0) indicating that the character is leaning forward as compared to the downhill plot. In either case, the character achieves a stable gait.*

to sloped terrain, steps, and moving platforms. These animations would be difficult to create using traditional key framing tools or spacetime optimization. In addition to varying the terrain, the animator can introduce external disturbances.

As a test case, we manually tuned a PD controller which balances through a foot placement strategy to walk on flat ground [2]. The same controller also works on a downhill slope of two degrees but fails on slopes of three degrees or higher. To walk down steeper slopes, one could readjust the target poses of the manual controller and its gains. However, our controller does this automatically. Given the reference output from the successful simulation on flat ground, our approach can derive controllers that successfully simulate the same motion on slopes of upto five degrees. This experiment demonstrates that controller adapts to new ground slopes even without any user intervention (see Figure 6).

[2]A SIMBICON controller produces stable walks on steeper slopes by departing from the original gait in a different way, namely by increasing step lengths and forward velocity[Yin et al. 2007].

Our controller was not designed to handle higher level motion planning tasks. For example, it does not know how to adapt a normal walk to walk up large steps. Applying our controller would result in the character stubbing its toe and falling. One approach to this problem is to build a set of primitive actions with our controller and to compose them together at runtime using the surrounding feedback. Our approach in this paper allows us to build such simple actions and we leave their automatic run-time composition for future work.

5.3 Composing Controllers

A library of motions can be used to derive many controllers, which can be composed together to produce complex actions. Similar ideas have been proposed in the past [Faloutsos et al. 2001; Yin et al. 2007; Sok et al. 2007], but the approach in this paper may make it simpler to derive such controls for a wider range of motions. Moreover, the new control produces high-quality motions that could be used along with kinematic motion graphs [Kovar et al. 2002; Arikan and Forsyth 2002]. In the video, we used the action graph in Figure 7 to jump between normal and marching styles including transitions between starting and stopping. In fact, even our simple walks are transitions between controllers for the left and right leg. Currently, the allowable transitions are specified manually. In the future, we hope to construct these action graphs automatically by precomputing optimal transitions based on character state as has been done for kinematic controllers [McCann and Pollard 2007; Treuille et al. 2007].

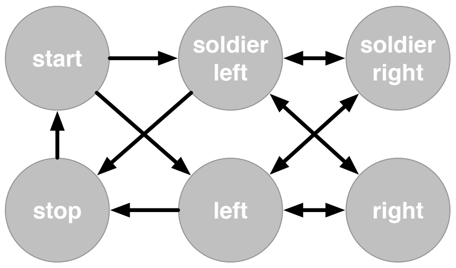

Figure 7: *An action graph consisting of controllers for normal and marching styles of walking. Currently, allowable transitions are found manually. An interesting area of future work is constructing action graphs automatically by precomputing optimal transitions based on character state.*

5.4 Experimental Setup

Physical Modelling and Simulation The motions were simulated using three different physical character models. The physical properties of these models are provided in Table 2. These models were constructed either to match the properties used from previous papers [Sok et al. 2007; Yin et al. 2007] or to match our motion capture subject. We set bone lengths to match the subjects dimensions and computed inertial properties by integrating a density over a polygonal human-like mesh [Winter 1990]. The joint stiffness gains are significantly lower than those used in previous controllers since most of the control effort is generated from the QP component of our controller. This makes the control less stiff which leads to more stable simulations. The motions were simulated in Open Dynamics Engine (ODE) with scenes created in DANCE [Shapiro et al. 2005].

Foot Clean up The raw mocap data is noisy, particularly for toes and ankles which are essential for stepping motions. We cleanup these motions to ensure that contact between the feet and the flat ground is consistent. First, a constant offset is applied to the ankle joints to align the feet. It is computed using a reference frame where the feet are assumed to be flat. Next motion edits are performed to constrain the character's toes to be flush against the ground when in contact. We use a "per-frame" kinematic fix with hierarchical B-spline smoothing, similar to the approach employed by [Lee and Shin 1999]. We manually annotate the segments of the motion with toe contacts and then compute the global transformation of the toe by projecting its position and orientation in the first frame of each segment onto the ground plane. A fast, analytic IK then adjusts the ankle, knee and hip joints minimally so that they connect with the positioned toe segments. The result of this per-frame adjustment is smoothed over the adjacent frames with a hierarchal B-spline. In the video, the displayed reference motions have all been fixed in this manner, except for the motions used in the previous work [Sok et al. 2007], which were used as is.

6 Conclusion

A control system that reproduces stylized locomotion has two major advantages. First, it can adapt motion-capture data to physically consistent dynamic environments. Second, it can produce large repertoires of simple actions. Both of these features are needed to create versatile characters in games and other interactive applications. In this work, we demonstrate such a control system for control of human stepping motions.

We show that automatic analysis of linear time-varying systems enables efficient precomputation of balance strategies. These strategies can be combined with a simple style feedback to simulate stylized human stepping motions. A possible additional application of our approach is combining it with existing controllers to improve their stability. For example, in the results section, we took a PD controller and extended its ability to walk on steeper slopes. Further investigation is needed to determine whether this additional feedback can improve stability of other controllers.

Large disturbances require higher level planning and more significant deviation from the reference motion. For example, a push might require taking several steps to retain balance. In this paper, we demonstrate manual composition of a few actions stabilized by our control system. In the future, higher-level planners should be developed to accomplish similar tasks automatically to improve robustness and respond to user input. Using a complex graph of simple actions, a higher-level planner could monitor the current state of the character, its environment, and the provided user input to simulate versatile characters in rich physically simulated environments.

7 Acknowledgments

Our work was aided by generous contributions from a number of people. Alexandre Megretski initiated this project by suggesting and explaining control of linear time-varying approximations. Russ Tedrake kept us up to date with publications in the vast robotics literature. Daniel Vlasic helped us acquire all of our motion data, while Kwang Won Sok, Manmyung Kim, and Jehee Lee shared theirs. And Tom Buehler volunteered his own time to help us with video production. This work was supported in part by the Singapore-MIT GAMBIT Game Lab, a National Science Foundation fellowship (2007043041), Pixar Animation Studios, and software donations from Autodesk and Adobe.

References

ABDALLAH, M.; GOSWAMI, A. 2005. A biomechanically motivated two-phase strategy for biped upright balance control. In *International Conference on Robotics and Automation (ICRA)*, 1996–2001.

ABE, Y., DA SILVA, M., AND POPOVIĆ, J. 2007. Multiobjective control with frictional contacts. In *Symposium on Computer Animation (SCA)*, 249–258.

ANITESCU, M., POTRA, F., AND STEWART, D., 1998. Time-stepping for threedimensional rigid body dynamics.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3 (July), 483–490.

ATKESON, C. G., AND MORIMOTO, J. 2003. Nonparametric representation of policies and value functions: A trajectory-based approach. In *Advances in Neural Information Processing Systems 15*, S. T. S. Becker and K. Obermayer, Eds. MIT Press, Cambridge, MA, 1611–1618.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH 94)*, ACM SIGGRAPH, Annual Conference Series, 23–34.

BARBIČ, J. 2007. *Real-time Reduced Large-Deformation Models and Distributed Contact for Computer Graphics and Haptics*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

BETTS, J. T. 2001. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, Philadelphia, PA.

BROTMAN, L. S., AND NETRAVALI, A. N. 1988. Motion interpolation by optimal control. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, 309–315.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum 27*, 2. In press.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, 251–260.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (July), 417–426.

FEATHERSTONE, R., AND ORIN, D. E. 2000. Robot dynamics: Equations and algorithms. In *International Conference on Robotics and Automation (ICRA)*, 826–834.

FULL, R. J., AND KODITSCHEK, D. E. 1999. Templates and anchors: Neuromechanical hypotheses of legged locomotion on land. *The Journal of Experimental Biology 202*, 3325–3332.

GILL, P. E., MURRAY, W., AND SAUNDERS, M. A. 1997. User's guide for SQOPT 5.3: A fortran package for large-scale linear and quadratic programming. Tech. Rep. NA 97–4, University of California, San Diego.

HODGINS, J. K., AND POLLARD, N. S. 1997. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 153–162.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, 71–78.

JACOBSON, D., AND MAYNE, D. 1970. *Differential Dynamic Programming*, 1st ed. Elsevier, New York.

KAJITA, S., KANEHIRO, F., KANEKO, K., FUJIWARA, K., HARADA, K., YOKOI, K., AND HIRUKAWA, H. 2003. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. 1644–1650.

KALMAN, R. E. 1960. Contributions to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana 5*, 102–119.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Transactions on Graphics 21*, 3 (July), 473–482.

LASZLO, J. F., VAN DE PANNE, M., AND FIUME, E. L. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 155–162.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, ACM SIGGRAPH, Annual Conference Series, 39–48.

LEWIS, F., ABDALLAH, C., AND DAWSON, D. M. 1993. *Control of Robot Manipulators*, 1st ed. Macmillan, New York.

LI, W., AND TODOROV, E. 2004. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation and Robotics*, 222–229.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (Aug.), 1071–1081.

MCCANN, J., AND POLLARD, N. 2007. Responsive characters from motion fragments. *ACM Transactions on Graphics 26*, 3 (July), 6:1–6:7.

MCGHEE, R. 1983. Vehicular legged locomotion. In *Advances in Automation and Robotics*, G. Saridis, Ed. JAI Press.

MIURA, H., AND SHIMOYAMA, I. 1984. Dynamic walk of a biped. *International Journal of Robotics Research 3*, 2, 60–74.

NGO, J. T., AND MARKS, J. 1993. Spacetime constraints revisited. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, 343–350.

POPOVIĆ, Z., AND WITKIN, A. P. 1999. Physically based motion transformation. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, ACM SIGGRAPH, Annual Conference Series, 11–20.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Computer Graphics (Proceedings of SIGGRAPH 2000)*, ACM SIGGRAPH, Annual Conference Series, 209–218.

PRATT, J., CARFF, J., DRAKUNOV, S., AND GOSWAMI, A. 2006. Capture point: A step toward humanoid push recovery. In *International Conference on Humanoid Robots*, 200–207.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, ACM SIGGRAPH, Annual Conference Series, 349–358.

RAIBERT, M. H. 1986. *Legged Robots That Balance*. MIT Press, Cambridge, MA.

SAFONOVA, A., HODGINS, J., AND POLLARD, N. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (Aug.), 514–521.

SHAPIRO, A., FALOUTSOS, P., AND NG-THOW-HING, V. 2005. Dynamic animation and control environment. In *Proceedings of Graphics Interface (GI)*, 61–70.

SHARON, D., AND VAN DE PANNE, M. 2005. Synthesis of controllers for sylized planar bipedal walking. In *International Conference on Robotics and Automation (ICRA)*, 2387–2392.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics 26*, 3 (July), 107:1–107:9.

STENGEL, R. F. 1994. *Optimal Control and Estimation*. Dover Books on Advanced Mathematics, New York, NY.

SULEJMANPASIĆ, A., AND POPOVIĆ, J. 2005. Adaptation of performed ballistic motion. *ACM Transactions on Graphics 24*, 1 (Jan.), 165–179.

TASSA, Y., EREZ, T., AND SMART, W. 2008. Receding horizon differential dynamic programming. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. MIT Press, Cambridge, MA, 1465–1472.

TEDRAKE, R. L. 2004. *Applied Optimal Control for Dynamically Stable Legged Locomotion*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. 2007. Near-optimal character animation with continuous control. *ACM Transactions on Graphics 26*, 3 (July), 7:1–7:7.

WIEBER, P. B. 2002. On the stability of walking systems. In *International Workshop on Humanoid and Human Friendly Robotics*, 1–7.

WINTER, D. A. 1990. *Biomechanics and Motor Control of Human Movement*, 2nd ed. John Wiley and Sons, Inc., New York.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, vol. 22, 159–168.

WOOTEN, W. L., AND HODGINS, J. K. 2000. Simulating leaping, tumbling, landing and balancing humans. *International Conference on Robotics and Automation (ICRA)*, 656–662.

YAMANE, K., AND NAKAMURA, Y. 2003. Dynamics filter - concept and implementation of online motion generator for human figures. *Robotics and Automation, IEEE Transactions on 19*, 3, 421–432.

YIN, K., CLINE, M., AND PAI, D. K. 2003. Motion perturbation based on simple neuromotor control models. In *Pacific Conference on Computer Graphics and Applications (PG)*, 445–449.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (July), 105:1–105:10.

YIN, K. K., 2008. personal communication.

ZORDAN, V. B., AND HODGINS, J. K. 1999. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Animation and Simulation*.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. In *Symposium on Computer Animation (SCA)*, 89–96.

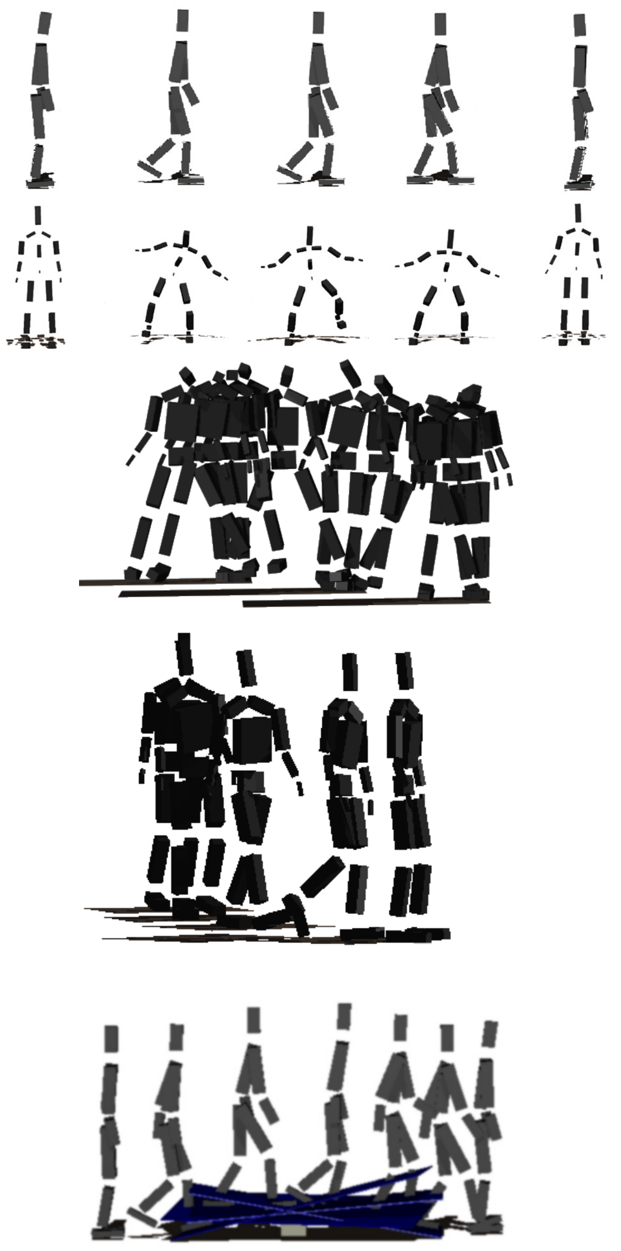

Figure 8: *Some examples of the styles of motion simulated by our controller: a backwards walk and a sumo maneuver. We can simulate reference motions in new environments, such as adapting a sideways stepping motion or a turning motion to walk down steps. The backwards walk is adapted to walk over a dynamic seesaw. The entire sequence is simulated creating a realistic motion that would be difficult to animate by hand, motion capture, or synthesize kinematically.*