Composite Control of Physically Simulated Characters

ULDARICO MUICO
University of Washington
JOVAN POPOVIĆ
University of Washington, Adobe Systems Incorporated
ZORAN POPOVIĆ
University of Washington

A physics-based control system that tracks a single motion trajectory produces high quality animations, but it does not recover from large disturbances that require deviating from this tracked trajectory. In order to enhance the responsiveness of physically simulated characters, we introduce algorithms that construct composite controllers that track multiple trajectories in parallel instead of sequentially switching from one control to the other. The composite controllers can blend or transition between different path controllers at arbitrary times according to the current system state. As a result, a composite control system generates both high quality animations and natural responses to certain disturbances. We demonstrate its potential for improving robustness in performing several locomotion tasks. Then we consolidate these controllers into graphs that allow us to direct the character in real time.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

Additional Key Words and Phrases: Character simulation, character control, physics-based character animation

ACM Reference Format:

Muico, U., Popovic, J., and Popovic, Z. 2011. Composite Control of Physically Simulated Characters. ACM Trans. Graph. 3, Article 16 (August 2011), 11 pages.
DOI = 10.1145/1966394.1966395
http://doi.acm.org/10.1145/1966394.1966395

This work was supported by the University of Washington Animation Research Labs (ARL), National Science Foundation (NSF) grant HCC-0811902, Intel, and Microsoft Research.
Authors' address: um@cs.washington.edu.

DOI 10.1145/1966394.1966395
http://doi.acm.org/10.1145/1966394.1966395

1. INTRODUCTION

Physically based simulation requires carefully crafted control systems to animate characters with agility and robustness found in nature. Deriving such systems from real data is one promising approach as it can be applied without modification to many locomotion skills including walking, stopping, turning and running. The control system tracks skill-specific trajectories to compute muscle forces that yield the same motion in physically based simulations. An immediate benefit of such approach is high quality of final animations as they look almost as real as the data they follow.

Such tracking controllers can preserve the style of the actor's locomotion, but the generated movements are usually monotonous reproductions. It is still a challenge to automatically find suitable motions that transition between different skills, sometimes in the presence of external forces and other disturbances. In order to enhance the realism of the simulation, the virtual character must deviate from rote tracking and adapt to changing conditions and environments. Furthermore biped locomotion is notoriously susceptible to falling, as the dynamical system must contend with the burden of underactuation.

One possibility is to have the controller learn from more data, so that a single control system leads to natural responses by automatically selecting which data to track. Here we present such an aggregate controller and the three inter-dependent processes needed to realize this goal for three-dimensional characters.

First, instead of tracking only one trajectory, our control system tracks multiple trajectories simultaneously, so that the character can respond better to unexpected situations. At any point in time, control forces are determined by automatically reweighting different actions. This allows for natural transitions between locomotion skills either by switching between their respective trajectories or by blending between them. The entire switching process is automatic, not authored.

Second, we show that this multi-trajectory composite controller can be constructed from graphs that connect unique motion trajectories. In contrast to the common graph traversal process where transitions are made only at the end of each trajectory clip, our composite controller switches and blends continuously through the branching structure of the graph, allowing it to transfer at any time, instead of just at the end of an edge. This creates a more pliable graph structure, leading to more responsive controllers.

Third, we show how our control system can accept high-level directives and integrate them in our composition process. With the greater availablity of possible paths, the combined action is more resilient to achieving the desired tasks. As we will see, the mapping from user commands to physical actions is ultimately a trade-

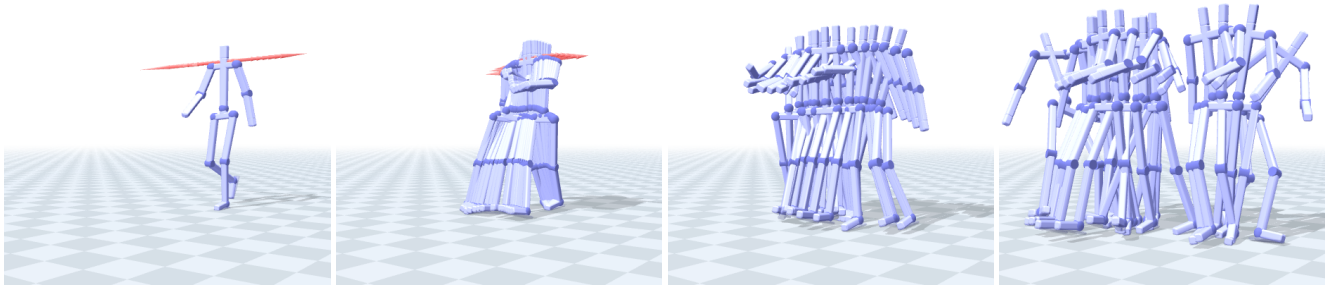

Fig. 1: *Different disturbances trigger different responses. From left to right, a character walking forward receives several sideway pushes of varying degree. A controller automatically determines an appropriate action.*

off between what is desirable at a high level and what is physically possible at the lower level.

Our composition method gives the virtual character more options to achieve long-term tasks. Besides the benefit of increased variability and unpredictibility in the character's behavior, our approach can be used to enhance the robustness of certain locomotion skills. Our control system can include latent responses that only emerge when they are compatible with unstable configurations. As a result, the controller can produce visually appealing recoveries in real time.

2. RELATED WORK

In computer animation, two general strategies have emerged for control of high-dimensional characters. In one category are state-based spring-damper systems inspired by the work of Raibert and Hodgins [1991]. Systems in this category have produced characters with a broad set of skills [Hodgins et al. 1995; Hodgins and Pollard 1997; Wooten and Hodgins 2000; Faloutsos et al. 2001; Yin et al. 2007; Coros et al. 2008] and perhaps even more notably characters that are resiliant to large pushes and disturbances [Yin et al. 2007; Raibert et al. 2008; Coros et al. 2009; Mordatch et al. 2010]. The other category includes tracking controllers that aim to reproduce motion trajectories within simulations [Laszlo et al. 1996; Zordan and Hodgins 2002; Sok et al. 2007; da Silva et al. 2008; Muico et al. 2009; Lee et al. 2010]. The use of motion capture in these systems has produced simulated characters whose motions are almost indistinguishable from motion capture. However, no technique in either category has delivered physically based characters that are both lifelike and robust.

The goal of our work is to widen the applicability of tracking controllers by forming primitives for more versatile control systems. Others have explored retuning control schemes to accommodate new tasks such as adapting running controllers from larger characters to smaller characters [Hodgins and Pollard 1997], modifying jumping controllers [Pollard and Behmaram-Mosavat 2000], prioritizing various objectives [de Lasa et al. 2010], or walking under various environmental constraints [Yin et al. 2008; Wang et al. 2010]. In contrast, we combine separate controllers as components of a composite control system.

Some systems concatenate components into sequences that achieve new tasks or recover from disturbances [Burridge et al. 1999; Faloutsos et al. 2001; Sok et al. 2007]. In our approach, components are not only executed in sequence but also in parallel so that several components may be active at any one time. Yin and collegues [2008] have observed that simple linear interpolation of SIMBICON [Yin et al. 2007] parameters can be used to adapt the walking controller. This observation was expanded upon to build a controller capable of taking different step lengths to avoid holes in the terrain [Coros et al. 2008]. However, simple linear interpolation of control parameters does not work on tracking controllers [da Silva et al. 2009] unless hand-crafted metrics are used to determine the blend weights [Sok et al. 2007; Erez and Smart 2007].

Our composition process rests on the observation that a nonlinear Bellman equation can be linearized under a change of variables [Fleming 1978; Todorov 2009b]. A similar idea also appears in the study of stochastic diffusion processes [Holland 1977]. Two recent works have relied on this observation to suggest composition as method for constructing control systems from simpler pieces [Todorov 2009a; da Silva et al. 2009]. Our paper delivers that method for three-dimensional characters with complex motions sourced from captured data. We make some key contributions.

First, we provide a framework that enables us to perform trajectory tracking control. Recent approaches solving general stochastic control problems [Todorov 2009b] have been shown to work well on small problems, but it is unclear how to apply them on interactive stylized characters due to a time-invariant formalism. We describe a control scheme that allows us to solve a simpler optimal control problem for following motion trajectories.

Secondly, previous works show composition under a considerable restriction: every individual component must originate from a common optimal control formulation [Todorov 2009a; da Silva et al. 2009]. Such motion trajectory optimization is difficult to generalize and renders these ideas inapplicable to controllers sourced from known motion data. Consequently, bipedal controllers have previously exhibited a limited range of behavioral styles and have been learned over short horizon times within the gait cycle. We address these shortcomings by developing a novel method that learns a control policy over a rich set of motions forming a cyclic graph.

Finally, we know of no other method that combines these ideas cohesively in an interactive control setting. We give a sensible solution to the interaction problem in accordance with the theme of optimal control.

3. CONTROL SYSTEM

Composition creates one universal controller by combining separate tracking controllers. For example, given a controller for walking straight ahead and a controller for stepping sideways, the universal controller combines them both, so that it responds to user directives or external pushes by blending or switching between these

two actions. Our composition procedure follows from the linearity of the Bellman equation, so we begin by showing how linear quadratic regulators (LQR) can be derived and used in that framework. Linearity allows us to combine cost functions of each component into one universal cost function. Then after proper numerical approximation, we can compute a composite control law that tracks multiple trajectories in parallel. However, LQR components are not resilient so we derive improved components that account for some nonlinearities as well as grazing and unilateral contacts.

3.1 System dynamics

For a system with n degrees of freedom, its state $\boldsymbol{x}$ consists of the pose $\boldsymbol{q} \in \mathbb{R}^n$ that describes the translational coordinates and joint orientations. It also consists of the corresponding velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ amounting to a state space, which we can define to be $\mathcal{X} = \mathbb{R}^n \times \mathbb{R}^n \equiv \mathbb{R}^{2n}$. Now, an essential element of this system is a feedback controller that follows a prescribed state trajectory $\bar{\boldsymbol{x}}(t) \in \mathcal{X}$. If the prescribed trajectory is consistent with some default flow

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{a}(\boldsymbol{x}(t)),$$

then the controller need not provide any corrective actions, and thus we refer to this default flow as the **passive dynamics**, or uncontrolled dynamics. In practice, control forces $\boldsymbol{u}(t) \in \mathcal{U}$ are needed to stabilize the system, and their influence is conveyed linearly in the equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{a}(\boldsymbol{x}(t)) + \boldsymbol{B}(\boldsymbol{x}(t))\,\boldsymbol{u}(t). \tag{1}$$

For an articulated body, the vector $\boldsymbol{a}$ represents the inertial and gravitational forces acting on the rigid bodies. Usually a system in contact with the environment induces a nonlinear coupling between $\dot{\boldsymbol{x}}$ and $\boldsymbol{u}$, but here we avoid this complication by augmenting the contact forces to $\boldsymbol{u}$ and treating them as actuation variables [Muico et al. 2009].

Such continuous-time models are useful in analyzing complex dynamical systems like ours, but they disregard the discrete-time nature of the simulated mechanics. Simulators use numerical integration to discretize the system dynamics and include complicated mechanisms to resolve constraint forces. We can model this simulated dynamics as a stochastic process, and in particular we can describe the passive dynamics as a transition probability distribution p over the state space $\mathcal{X}$. Using Euler integration with step size h, we take it to be the Gaussian distribution

$$p(\,\cdot \mid \boldsymbol{x}\,) = \mathcal{N}\big(\boldsymbol{x} + h\boldsymbol{a}(\boldsymbol{x}),\; h\boldsymbol{\Sigma}(\boldsymbol{x})\big). \tag{2}$$

Here, we assume that the covariance matrix for this distribution spans the control subspace, which we can express as

$$\boldsymbol{\Sigma}(\boldsymbol{x}) = \boldsymbol{B}(\boldsymbol{x})\,\boldsymbol{R}^{-1}\boldsymbol{B}(\boldsymbol{x})^{\mathsf{T}}. \tag{3}$$

The matrix $\boldsymbol{R}$ is symmetric positive definite and signifies our degree of belief in the passive dynamics. Note that since our system is underactuated, this covariance matrix is singular with a rank no greater than n. The passive distribution serves as a prior distribution, and now the task at hand is to find a posterior probability distribution π suitable for tracking a sequence of desired states $\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_N$.

3.2 Trajectory tracking

One can approach the trajectory-following problem in various ways, but for our purposes we develop the tracking problem upon the foundation of stochastic optimal control theory. In doing so, we establish certain recurring quantities and then construct primitives required for the main problem.

While the primary objective of optimal tracking is to minimize the deviation of the system state from a nominal trajectory, it also minimizes the exerted control, so that the system does not aggressively stray from the flow of the passive dynamics. We can express this problem recursively in a **Bellman equation**

$$v_k(\boldsymbol{x}) = \min_{\pi} \{\, \ell_k(\boldsymbol{x}, \pi) + \mathrm{E}_{\boldsymbol{y}\sim\pi}\left[v_{k+1}(\boldsymbol{y})\right] \}\,, \tag{4}$$

in which the **cost-to-go** functions v_k accumulate penalties ℓ_k along state transitions governed by the process $\boldsymbol{x}_{k+1} \sim \pi(\,\cdot \mid \boldsymbol{x}_k\,)$. We choose a penalty function of the form

$$\ell_k(\boldsymbol{x}, \pi) = c_k(\boldsymbol{x}) + \mathrm{KL}\left[\,\pi \parallel p\,\right], \tag{5}$$

where the tracking cost c_k measures the deviation of the state from the desired state $\bar{\boldsymbol{x}}_k$, while the second penalty is the Kullback-Leibler (KL) divergence

$$\mathrm{KL}\left[\,\pi \parallel p\,\right] = \int_{\mathcal{X}} \pi(\,\boldsymbol{y} \mid \boldsymbol{x}\,) \log \frac{\pi(\,\boldsymbol{y} \mid \boldsymbol{x}\,)}{p(\,\boldsymbol{y} \mid \boldsymbol{x}\,)}\,\mathrm{d}\boldsymbol{y}.$$

This is a natural measure for the discrepancy between the controlled and the passive dynamics, and it serves to subdue any highly aggressive behavior.

This class of optimal control problems admits an analytical solution, and if we can solve for the cost-to-go functions, then it follows that the transition probability for the **controlled dynamics** is given by [Todorov 2009b]

$$\pi_{k+1}(\,\cdot \mid \boldsymbol{x}\,) = \frac{p(\,\cdot \mid \boldsymbol{x}\,)\, e^{-v_{k+1}(\cdot)}}{\int_{\mathcal{X}} p(\,\boldsymbol{y} \mid \boldsymbol{x}\,)\, e^{-v_{k+1}(\boldsymbol{y})}\,\mathrm{d}\boldsymbol{y}}. \tag{6}$$

In general, there is no closed-form solution for this expression, but problems with quadratic cost functions in fact yield controlled dynamics having a Gaussian distribution. To see this, suppose that the cost-to-go function at the next time step $k+1$ is quadratic

$$\begin{aligned} v_{k+1}(\boldsymbol{y}) = \bar{v}_{k+1} &+ \boldsymbol{g}_{k+1}^{\mathsf{T}}\,(\boldsymbol{y} - \bar{\boldsymbol{x}}_{k+1}) \\ &+ \frac{1}{2}\,(\boldsymbol{y} - \bar{\boldsymbol{x}}_{k+1})^{\mathsf{T}}\boldsymbol{H}_{k+1}\,(\boldsymbol{y} - \bar{\boldsymbol{x}}_{k+1}), \end{aligned} \tag{7}$$

whose parameters $\bar{v}_{k+1}$, $\boldsymbol{g}_{k+1}$ and $\boldsymbol{H}_{k+1}$ are referred to respectively as the **bias**, **gradient** and **Hessian** of the function at $\bar{\boldsymbol{x}}_{k+1}$. Since the passive distribution is Gaussian in Equation 6, we can identify both the mean and covariance of the controlled distribution by completing the square in the exponent. This gives us the solution

$$\begin{aligned} \pi_{k+1}(\,\cdot \mid \boldsymbol{x}\,) = \mathcal{N}\big(&\boldsymbol{x} + h\boldsymbol{a}(\boldsymbol{x}) + h\boldsymbol{B}(\boldsymbol{x})\,\boldsymbol{u}_k(\boldsymbol{x}), \\ &h\boldsymbol{B}(\boldsymbol{x})\,\boldsymbol{R}_k(\boldsymbol{x})^{-1}\boldsymbol{B}(\boldsymbol{x})^{\mathsf{T}}\big), \end{aligned} \tag{8}$$

whose covariance and mean can be computed using

$$\boldsymbol{R}_k(\boldsymbol{x}) = \boldsymbol{R} + h\boldsymbol{B}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{H}_{k+1}\,\boldsymbol{B}(\boldsymbol{x}), \tag{9a}$$

$$\boldsymbol{u}_k(\boldsymbol{x}) = -\boldsymbol{R}_k(\boldsymbol{x})^{-1}\boldsymbol{B}(\boldsymbol{x})^{\mathsf{T}}\,\boldsymbol{\nabla} v_{k+1}(\boldsymbol{x} + h\boldsymbol{a}(\boldsymbol{x})). \tag{9b}$$

Observe that this new distribution is slightly more peaked than the passive distribution, and such diminished covariance is a reflection of the controller's role in reducing the uncertainty in the system state. Moreover, we find that Equation 9b is the same as the linear quadratic tracking control law, but we derived it in a stochastic framework needed for our composition procedure.

Notice that this solution assumes that we have in possession a cost-to-go v_{k+1} satisfying Equation 7. Later in Section 4, we give an algorithm to compute such quadratic functions.

3.3 Composite trajectory tracking

A controller constructed out of the foregoing prescription specializes in following a single trajectory. Now we show how to combine such trajectory controllers and to track several trajectories in parallel. Composition problems of this kind have been established in recent works [Todorov 2009a; da Silva et al. 2009], albeit in time-invariant and continuous-time settings. Whereas they show a top-down approach to derive primitive controllers, we instead take a bottom-up approach that consolidates *arbitrary* controllers. As a result, the universal controller can exhibit greater variety of responses when its components individually contribute distinctive behavior.

We shall refer to each **component** by its index and label its associated quantities appropriately with superscripts. Thus for each component i, we have some cost-to-go function v^i_{k+1}. We combine all such cost functions in a soft-min manner into one universal cost

$$v_{k+1}(\boldsymbol{y}, \boldsymbol{w}) = -\log \sum_{i=1}^{m} w^i \, e^{-v^i_{k+1}(\boldsymbol{y})}, \tag{10}$$

where $\boldsymbol{w}$ is a non-negative vector that indicates each component's influence. We show in Appendix A.2 that we can phrase a new optimal control problem satisfying Equation 4 by having a combined tracking cost

$$c_k(\boldsymbol{x}, \boldsymbol{w}) = \log \sum_{i=1}^{m} \alpha^i_k(\boldsymbol{x}, \boldsymbol{w}) \, e^{c^i_k(\boldsymbol{x})}, \tag{11}$$

where the component costs c^i_k are weighted by factors

$$\alpha^i_k(\boldsymbol{x}, \boldsymbol{w}) = \frac{w^i e^{-v^i_k(\boldsymbol{x})}}{\sum_{j=1}^{m} w^j e^{-v^j_k(\boldsymbol{x})}}, \tag{12}$$

which favor the cheaper cost-to-go. Later we use this cost function to measure the performance of our controllers.

The solution to this composite problem follows straightforwardly from the linearity of the Bellman equation [Todorov 2009a]. From there, we find that the optimal probability distribution is a convex combination of the individual optimal distributions

$$\pi_{k+1}(\cdot \mid \boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^{m} \beta^i_k(\boldsymbol{x}, \boldsymbol{w}) \, \pi^i_{k+1}(\cdot \mid \boldsymbol{x}), \tag{13}$$

where each individual solution π^i_{k+1} is weighted by the factor

$$\beta^i_k(\boldsymbol{x}, \boldsymbol{w}) = \frac{w^i \, e^{c^i_k(\boldsymbol{x}) - v^i_k(\boldsymbol{x})}}{\sum_{j=1}^{m} w^j \, e^{c^j_k(\boldsymbol{x}) - v^j_k(\boldsymbol{x})}}. \tag{14}$$

For our particular tracking problem, this implies that the optimal state transition is described by a mixture of Gaussian distributions. Unlike the single-tracking case where the mean and the maximum of a Gaussian solution obviously coincide, an optimal control law like Equation 9b is not well-defined for this mixture. This poses a problem because we require a concrete control signal to drive a deterministic simulation system.

For this composite problem, we reason that the correct actuations correspond to those of the *maximum* of the probability distribution. In the common situation where we combine vastly dissimilar components, this strategy can reduce the crosstalk between competing components and thus deliver coherent actions. We therefore proceed to compute the control law $\boldsymbol{u}_k(\boldsymbol{x}, \boldsymbol{w})$ by solving the non-convex optimization problem

$$\begin{aligned} \text{minimize} \quad & -\log \pi_{k+1}(\, \boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}\,) \\ \text{over all} \quad & \boldsymbol{y}, \boldsymbol{u} \\ \text{subject to} \quad & \boldsymbol{y} = \boldsymbol{x} + h\boldsymbol{a}(\boldsymbol{x}) + h\boldsymbol{B}(\boldsymbol{x})\, \boldsymbol{u}. \end{aligned} \tag{15}$$

This is an unwieldy computation in general, so we choose to approximate the solution for our problem with Gaussian distributions. If the step h is small, we can assume a locally convex distribution near the *expected* state

$$\tilde{\boldsymbol{y}}_k(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{x} + h\boldsymbol{a}(\boldsymbol{x}) + h\boldsymbol{B}(\boldsymbol{x})\, \tilde{\boldsymbol{u}}_k(\boldsymbol{x}, \boldsymbol{w}) \tag{16}$$

arising from the expected control

$$\tilde{\boldsymbol{u}}_k(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^{m} \beta^i_k(\boldsymbol{x}, \boldsymbol{w})\, \boldsymbol{u}^i_k(\boldsymbol{x}). \tag{17}$$

This state serves as an initial point about which we compute the gradient of the objective function within the control subspace. Differentiation gives the gradient vector

$$\boldsymbol{\lambda}_k(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^{m} \alpha^i_{k+1}(\tilde{\boldsymbol{y}}_k, \boldsymbol{w})\, \boldsymbol{\lambda}^i_k(\boldsymbol{x}, \boldsymbol{w}), \tag{18}$$

where each component i contributes the vector

$$\boldsymbol{\lambda}^i_k(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{R}^i_k(\boldsymbol{x}) \left(\tilde{\boldsymbol{u}}_k(\boldsymbol{x}, \boldsymbol{w}) - \boldsymbol{u}^i_k(\boldsymbol{x})\right). \tag{19}$$

Meanwhile the Hessian at the expected state is

$$\begin{aligned} \boldsymbol{\Lambda}_k(\boldsymbol{x}, \boldsymbol{w}) &= \boldsymbol{\lambda}_k(\boldsymbol{x}, \boldsymbol{w})^{\mathsf{T}} \boldsymbol{\lambda}_k(\boldsymbol{x}, \boldsymbol{w}) \\ &+ \sum_{i=1}^{m} \alpha^i_{k+1}(\tilde{\boldsymbol{y}}_k, \boldsymbol{w}) \left(\boldsymbol{R}^i_k(\boldsymbol{x}) - \boldsymbol{\lambda}^i_k(\boldsymbol{x}, \boldsymbol{w})\, \boldsymbol{\lambda}^i_k(\boldsymbol{x}, \boldsymbol{w})^{\mathsf{T}}\right). \end{aligned} \tag{20}$$

In the case that this matrix is not positive definite, one can simply perform a gradient descent until $\tilde{\boldsymbol{u}}$ enters a locally convex region. At this point, Newton's method gives an approximate solution

$$\boldsymbol{u}^{(1)}(\boldsymbol{x}, \boldsymbol{w}) = \tilde{\boldsymbol{u}}_k(\boldsymbol{x}, \boldsymbol{w}) - \boldsymbol{\Lambda}_k(\boldsymbol{x}, \boldsymbol{w})^{-1} \boldsymbol{\lambda}_k(\boldsymbol{x}, \boldsymbol{w}). \tag{21}$$

Observe that in the case $m = 1$, we conveniently recover the linear quadratic regulator of Equation 9b. In fact, this solution reduces to the continuous-time control law [Todorov 2009a; da Silva et al. 2009] as the time step h vanishes.

We can optionally repeat this process as in sequential quadratic programming (SQP) to find more accurate solutions $\boldsymbol{u}^{(2)}, \boldsymbol{u}^{(3)}, \ldots$. One simply replaces the initial guess $\tilde{\boldsymbol{u}}$ with the preceding iterate. Usually it suffices to accept the control law in Equation 21, especially when the complexity of the whole control system becomes an obstacle.

3.4 Constrained control

The optimization in Problem 15 may be adequate for typical locomotion skills, but it still relies on simplified passive dynamics in Equation 2. Like the primitive control law of Equation 9b, this solution admits unphysical contact forces such as those that result in

ground penetration or sliding. One way to resolve such violations is by introducing an intermediary stabilization stage that corrects the control predictions [Muico et al. 2009]. We too take this approach, but it might be prudent to take a step further and enforce some additional safety measures that account for the deficiencies in our model and ensure that the solution remain close to being physically valid.

First, we enforce unilateral contact constraints on the predicted contact forces $\boldsymbol{\chi}$. This typically involves a linear complementarity condition

$$\boldsymbol{\chi} \geq \mathbf{0}, \quad \mathrm{D}\boldsymbol{r}(\tilde{\boldsymbol{q}}_{k+1})\,\dot{\boldsymbol{q}} \geq \mathbf{0}, \tag{22a}$$

$$\boldsymbol{\chi}^{\mathsf{T}}\,\mathrm{D}\boldsymbol{r}(\tilde{\boldsymbol{q}}_{k+1})\,\dot{\boldsymbol{q}} = 0, \tag{22b}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the unknown generalized velocity vector, $\boldsymbol{r}$ measures the contact separation as a function of the pose, and $\mathrm{D}\boldsymbol{r}$ is the Jacobian of $\boldsymbol{r}$. The vector $\tilde{\boldsymbol{q}}_{k+1} \in \mathbb{R}^n$ is an estimation of the next pose, which we can infer from Equation 21. The complementarity constraint of Equation 22b is a nonlinear relationship that is difficult to enforce. At this stage, we relax this condition by focusing only on the inequality constraints. Since we are looking for a simple prediction, we can defer the enforcement of complementarity later during the simulation stage.

Next, we constrain limbs that are not in contact with the ground. As we anticipate certain contact configurations from the reference trajectory, we can logically discern swinging limbs that may scuff the ground too early. These events often lead to stumbling, so we reduce them by introducing another linear constraint

$$\mathbf{0} \leq \mathrm{diag}(\boldsymbol{\rho}_k)\,\boldsymbol{s}(\tilde{\boldsymbol{q}}_{k+1}) + h\,\mathrm{D}\boldsymbol{s}(\tilde{\boldsymbol{q}}_{k+1})\,\dot{\boldsymbol{q}}, \tag{23}$$

where $\boldsymbol{s}$ consists of distances from some limbs to the ground as a function of the pose, and the parameter ρ_k^j provides damping that controls the rate of separation for body point j. We found the following damping parameter to work well with many motions

$$\rho_k^j = \kappa + (1-\kappa)\,e^{-\frac{1}{2}\,(\sigma_k^j/\bar{\sigma})^2}, \tag{24}$$

where σ_k^j is the current tangential speed of the point j, and $\kappa = 0.1$ and $\bar{\sigma} = 1\,\mathrm{m/s}$. We can activate this constraint only when the separation is small and when the separation velocity is attractive. In this way, we reduce high-velocity impacts by adjusting torques to keep such critical bodies afloat. This strategy is useful particularly for a reduced-coordinate system like ours having no other special treatment of its end-effectors.

Finally, we constrain allowed control forces. Although the KL divergence provides a similar function, its role is primarily to regularize the optimal control problem. As a result, the character may still react too stiffly in the face of disturbances. We prevent this scenario with torque limits.

Since we approximated the solution to Problem 15 with the composite control law of Equation 21, we likewise approximate our new problem as a quadratic program as follows.

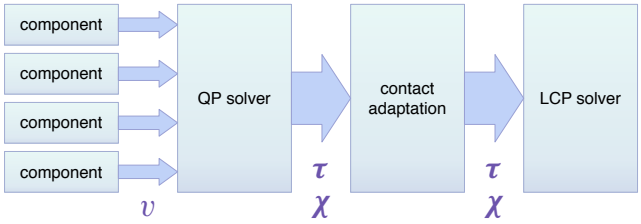

Fig. 2: Forward dynamics. *At runtime, components contribute their cost-to-go v to be combined in a quadratic program. The optimization computes predicted controls to be adapted using a more precise contact model.*

$$\begin{aligned}
\text{minimize} \quad & \begin{bmatrix}\boldsymbol{\tau}\\ \boldsymbol{\chi}\end{bmatrix}^{\mathsf{T}} \left(\boldsymbol{\lambda}_k(\boldsymbol{x}_k,\boldsymbol{w}_k) - \boldsymbol{\Lambda}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)\,\tilde{\boldsymbol{u}}(\boldsymbol{x}_k,\boldsymbol{w}_k)\right) \\
& \qquad + \frac{1}{2}\begin{bmatrix}\boldsymbol{\tau}\\ \boldsymbol{\chi}\end{bmatrix}^{\mathsf{T}} \boldsymbol{\Lambda}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)\begin{bmatrix}\boldsymbol{\tau}\\ \boldsymbol{\chi}\end{bmatrix} \\
\text{over all} \quad & \dot{\boldsymbol{q}} \in \mathbb{R}^n,\ \boldsymbol{\tau} \in \mathbb{R}^{n-6},\ \boldsymbol{\chi} \in \mathbb{R}^{n_c} \\
\text{subject to} \quad & \boldsymbol{M}(\boldsymbol{q}_{k+1})\,(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_k) \\
& \qquad = \boldsymbol{f}(\boldsymbol{q}_{k+1},\dot{\boldsymbol{q}}_k) + \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\tau}\end{bmatrix} + \mathrm{D}\boldsymbol{r}(\boldsymbol{q}_{k+1})^{\mathsf{T}}\boldsymbol{\chi}, \\
& \mathbf{0} \leq \boldsymbol{\chi}, \\
& \mathbf{0} \leq \mathrm{D}\boldsymbol{r}(\boldsymbol{q}_{k+1})\,\dot{\boldsymbol{q}}, \\
& \mathbf{0} \leq \mathrm{diag}(\boldsymbol{\rho}_k)\,\boldsymbol{s}(\boldsymbol{q}_{k+1}) + h\,\mathrm{D}\boldsymbol{s}(\boldsymbol{q}_{k+1})\,\dot{\boldsymbol{q}}, \\
& \boldsymbol{\tau}_{\mathrm{low}} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\mathrm{high}}.
\end{aligned} \tag{25}$$

Here, the objective function is chosen such that its minimum coincides with the unconstrained control law of Equation 21. The matrix $\boldsymbol{M}$ is the inertia of the articulated body, while $\boldsymbol{f}$ consists of fictitious forces, as well as gravitational and other external forces. The result of this optimization replaces Equation 21, which in turn replaces the primitive control law of Equation 9b. It serves as a prediction that is subsequently adapted to account for actual contact configurations, frictional constraints and disturbances.

3.5 Contact adaptation

The optimization in Problem 25 may be sufficient for undisturbed locomotion where the character's environment closely resembles the conditions assumed by the prescribed trajectories. In this case, no further corrections are needed, and we can simply evolve the system forward using Euler's method. When a more precise simulation dictates that contacts fully adhere to complementarity constraints in Equation 22b, it is possible to adapt our solution to meet the requirement. We achieve this by constructing a linear controller that embodies the response predicted by the quadratic program. We follow the approach of previous work [Muico et al. 2009], whereby a controller is fed into a linear complementarity problem (LCP) solver to maintain some accelerations.

This contact adaptation stage and the subsequent simulation completes the computation of the forward dynamics. This whole process is summarized in Figure 2.

4. MOTION GRAPH

A crucial part of the composition process is identifying which trajectories are composable. Given the cost-to-go function of every component, we can, in principle, compose all possible controllers at each instant to arrive at a highly coordinated action. Clearly this approach is not practical, and we need to limit our search for compatible trajectories. Therefore we choose to connect our trajectories in a graph. This structure allows us to manually specify neighboring motions and to anticipate various locomotion skills.

4.1 Multiplicity

Suppose a trajectory T connects to other trajectories $\mathsf{S}^1, \ldots, \mathsf{S}^m$ as shown in Figure 3(a). The policy for T is a sequence of control laws $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ that must be optimal in some sense for each of the m possible outcomes. In order to encode such prospective motions into this sequence of controllers, the information must propagate backwards from all these outcomes. Rather than accumulate such responses into a single control law given by Equation 9b, we can instead solve for separate control laws corresponding to all the possible outcomes S^i.

As shown in Figure 3(b), we propose that the trajectory T be copied with **multiplicity** m, the number of outcomes branching from T. Each copy is associated with the same original state trajectory, but its control law is tailored for a specific future. At any point during the simulation, the composite controller is therefore blending the components that correspond to all possible outcomes presented by a set of trajectories as shown in Figure 3(c). This allows a greater continuity of trajectories and a significantly more flexible graph structure. Furthermore, the composition reduces instabilities of the single point of transition in standard switching graphs.

4.2 Dynamic programming

Now we use this organization to learn the controllers for each component. As described in Section 3, these controllers depend entirely on the parameters of cost-to-go functions v_k^i. Using a dynamic programming approach, we can in fact compute cost-to-go functions for finite-horizon problems with *linear* dynamics. The solution to these problems are given by algebraic Riccati equations (ARE) that determine the gradient and Hessian of Equation 7. Refer to Appendix A.1 for a precise description of these equations.

In order to apply this method, it suffices that the flow $\boldsymbol{a}$ of the passive dynamics be approximated as a linear function of the state $\boldsymbol{x}$, and that the covariance matrix be fixed. In this way, we get a simplified passive dynamics that is Gaussian

$$p_k(\cdot \mid \boldsymbol{x}) = \mathcal{N}\big(\bar{\boldsymbol{y}}_k + \boldsymbol{A}_k\,(\boldsymbol{x} - \bar{\boldsymbol{x}}_k)\,,\; h\boldsymbol{B}_k\,\boldsymbol{R}^{-1}\boldsymbol{B}_k^{\mathsf{T}}\big). \tag{26}$$

Notice that this probability distribution is time-varying with parameters $\bar{\boldsymbol{y}}_k$, $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ that depend on the choice of the finite differencing scheme. To compute these parameters, suppose we are given the reference state $\bar{\boldsymbol{x}}_k$ and also a reference control $\bar{\boldsymbol{u}}_k$, which we can obtain from the method of space-time optimization. We can then construct the Taylor expansion of the simulation function $\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x} + h\boldsymbol{a}(\boldsymbol{x}) + h\boldsymbol{B}(\boldsymbol{x})\,\boldsymbol{u}$, which to first order is

$$\begin{aligned}\tilde{\boldsymbol{\psi}}_k(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{\psi}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) &+ \mathrm{D}_{\boldsymbol{x}}\boldsymbol{\psi}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)\,(\boldsymbol{x} - \bar{\boldsymbol{x}}_k) \\ &+ \mathrm{D}_{\boldsymbol{u}}\boldsymbol{\psi}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k)\,(\boldsymbol{u} - \bar{\boldsymbol{u}}_k)\,.\end{aligned}$$

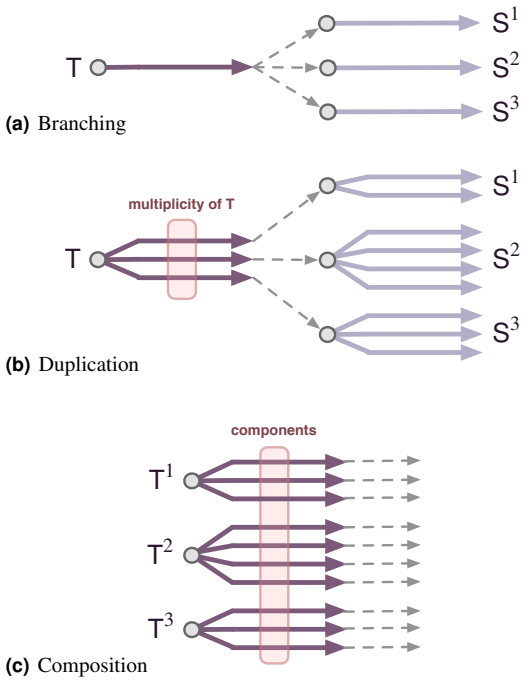

Fig. 3: Graph construction. **(a)** *A trajectory* T *connects to* 3 *prospective trajectories.* **(b)** *A trajectory* T *has multiplicity* 3*, and each of the three copies will have a different policy synthesized depending on its outcome.* **(c)** *At runtime, a composite controller with a working set of* 3 *trajectories is actually evaluating many components depending on the total number of possible outcomes.*

Now, the mean of the passive distribution p is approximately the linear function $\tilde{\boldsymbol{\psi}}_k(\boldsymbol{x}, \boldsymbol{0})$, from which we identify the coefficients

$$\bar{\boldsymbol{y}}_k = \bar{\boldsymbol{x}}_k + h\boldsymbol{a}(\bar{\boldsymbol{x}}_k), \tag{27a}$$
$$\boldsymbol{A}_k = \mathrm{D}_{\boldsymbol{x}}\boldsymbol{\psi}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k), \tag{27b}$$
$$\boldsymbol{B}_k = \boldsymbol{B}(\bar{\boldsymbol{x}}_k). \tag{27c}$$

With these quantities, we can finally solve for the parameters of the cost-to-go using the ARE.

Because of the simplification of the passive dynamics, this solution is valid only locally around the reference trajectory. Since our goal is mainly to reproduce this fixed sequence of states, we find that such a crude approximation can adequately capture the dynamics along the path. Still, our control system must provide better coverage for real-world operation, and subsequently we explore dynamic programming in the composite setting.

4.3 Network dynamic programming

As our mechanical system is nonlinear and our trajectories interconnected in a cyclic graph, our main problem is ideally a nonlinear infinite-horizon problem. We resort to finding an approximation to the cost-to-go function by using a novel iterative process.

Suppose that for each component i of a trajectory T, we have a terminal cost function v_N^i that is quadratic. If we linearize the dynamics along the trajectory using Equation 26, then we can solve the Riccati equations backwards from the terminal state and obtain

a sequence of cost-to-go functions for component i. The problem remains in finding an appropriate function v_N^i.

Since the component i is not actually terminal but is connected to trajectory S^i, we reason that v_N^i is the cost-to-go function at the *initial* state of S^i. Note that this outcome trajectory itself has a nontrivial multiplicity, as it is associated with multiple cost-to-go functions corresponding to its own outcomes. Consequently, we combine these functions at this nexus in order to determine an effective cost-to-go v_N^i. Just as we found a quadratic approximation of the mixture distribution in Problem 15, we can likewise find a quadratic approximation to Equation 10. To second order, the terminal cost-to-go at the state $\bar{\boldsymbol{x}}_N$ is given by

$$\bar{v}_N^i = v_1(\bar{\boldsymbol{x}}_N, \boldsymbol{w}), \tag{28a}$$

$$\boldsymbol{g}_N^i = \sum_{j=1}^{m^i} \alpha_1^j(\bar{\boldsymbol{x}}_N, \boldsymbol{w})\, \boldsymbol{\nabla} v_1^j(\bar{\boldsymbol{x}}_N), \tag{28b}$$

$$\boldsymbol{H}_N^i = \boldsymbol{g}_N^i \boldsymbol{g}_N^{i\,\mathsf{T}} + \sum_{j=1}^{m^i} \alpha_1^j(\bar{\boldsymbol{x}}_N, \boldsymbol{w}) \left(\boldsymbol{H}_1^j - \boldsymbol{\nabla} v_1^j(\bar{\boldsymbol{x}}_N) \boldsymbol{\nabla} v_1^j(\bar{\boldsymbol{x}}_N)^{\mathsf{T}}\right), \tag{28c}$$

where the index j refers to one of the m^i components of S^i. Typically the trajectory S^i is not aligned with T and must be transformed prior to the evaluation of Equation 28. We discuss in Appendix B.1 that this is equivalent to transforming the parameters of the cost-to-go function.

In the exceptional case that the effective Hessian $\boldsymbol{H}_N^i$ is not positive definite, one possible course of action is to simply ignore all the outer product terms culpable in Equation 28c. Alternatively, the more proper treatment would be to modify the reference states so as to smoothy transition into the outcome S^i. Since we are simply looking for a terminal cost, the former approach is sufficient to handle malformed compositions.

Notice that it may be unnecessary to compute the biases $\bar{v}_k$ of the quadratic value functions, as these values can be absorbed into the unspecified component weights $\boldsymbol{w}$. In fact, these weights can be manually assigned to indicate preferred trajectories or to suppress certain behavior. As we will see in the next section, the arbitrariness of such an assignment can be removed by applying a high-level control scheme.

We summarize the dynamic programming algorithm below. By completing this process, we have effectively created local controllers whose futures are informed by the structure of the graph, thus improving their performance over controllers constructed independently of the graph structure.

Algorithm 1 Dynamic programming

```
Initialize all biases, gradients and Hessians to zero.
repeat
  for all trajectories T do
    for all outcome trajectories S of T do
      Evaluate the terminal parameters of Equation 28 using
      the current parameters for S.
      Solve ARE backwards along T, updating the control
      laws of the component corresponding to S.
    end for
  end for
until the biases, gradients and Hessians converge.
```

4.4 Traversal

At runtime, we must quickly decide which trajectories are composable. This is straightforward when we are already evaluating a **working set** $\mathcal{T} = \{\mathsf{T}^1, \mathsf{T}^2, \dots\}$ consisting of trajectories that have yet to be completely traversed. At some point, the control system must choose a new set $\mathcal{T}$ once it exhausts the current one. One possibility is to choose all trajectories branching from the current set, but this strategy would lead to an exponential growth of the set. Instead we implement the following strategy.

(1) Find the component i with the largest factor α^i evaluated at the current state.

(2) Identify the trajectory T associated with the component i.

(3) Choose as the new working set all outcome trajectories $\{\mathsf{S}^1, \mathsf{S}^2, \dots\}$ branching from T.

Note that the working set of trajectories must satisfy the requirement that their control subspaces be identical at any instant. Since we are treating contacts as actuators in our control prediction, this requirement implies that the predicted contact configurations be the same in any working set. For example, we cannot compose the flight phase of a running trajectory with the double-support configuration of a standing trajectory. This restriction further limits our choice of the working set during phase transitions.

5. HIGH-LEVEL CONTROL

The component weights $\boldsymbol{w}$ that determine the low-level dynamics have thus far been arbitrary. At a higher-level, these weights provide extra degrees of freedom that we can exploit to perform high-level tasks such as responding to user directives.

We augment the low-level state vector $\boldsymbol{x}$ with a task parameter $\boldsymbol{\theta}$ to form a higher-level state $(\boldsymbol{x}, \boldsymbol{\theta})$. For example, the task can be the desired speed or direction of motion. The state transitions are assumed to be distributed as

$$\boldsymbol{x}_{k+1} \sim \pi_{k+1}(\cdot \mid \boldsymbol{x}_k, \boldsymbol{w}_k), \qquad \boldsymbol{\theta}_{k+1} \sim \Pi_{k+1}(\cdot \mid \boldsymbol{x}_k, \boldsymbol{\theta}_k),$$

where π_{k+1} is given by Equation 6, and Π_{k+1} describes the user behavior. Then we can regard $\boldsymbol{w}_k$ as a high-level action whose policy we must determine in order to achieve high-level control goals.

Consider the problem of finding an optimal policy $\boldsymbol{w}_k^\star$ that coincides with the Bellman equation

$$V_k(\boldsymbol{x}_k, \boldsymbol{\theta}_k) = \min_{\boldsymbol{w}} \big\{\, C_k(\boldsymbol{x}_k, \boldsymbol{\theta}_k) + \mathrm{KL}\left[\, \boldsymbol{w} \parallel \bar{\boldsymbol{w}}_k \,\right] + \rho\, \mathrm{E}\left[V_{k+1}(\boldsymbol{x}_{k+1}, \boldsymbol{\theta}_{k+1})\right] \big\}, \tag{29}$$

where $0 \le \rho < 1$ is a discount factor. Here the new cost function C_k expresses the proximity of the system to achieving the high-level task, while the KL divergence measures the deviation of the action from some default distribution $\bar{\boldsymbol{w}}_k$. For example, suppose that $\bar{\boldsymbol{w}}_k$ is uniform. Then this cost term penalizes those strongly biased weights $\boldsymbol{w}$ having low entropy. Such information-laden actions are seen to be aggressive, while disordered actions can be regarded favorably as being effortless or lazy.

Due to the complexity of this high-level problem, we find it too expensive to seek a policy $\boldsymbol{w}_k^\star(\boldsymbol{x}, \boldsymbol{\theta})$ for every trajectory and at every instant in time. Instead, we settle for solving a single time-invariant policy weighing the outcomes $\mathsf{S}^1, \dots, \mathsf{S}^l$ branching from a working set $\mathcal{T}$ of trajectories. That is, for every initial state $(\boldsymbol{x}_1, \boldsymbol{\theta}_1)$ that

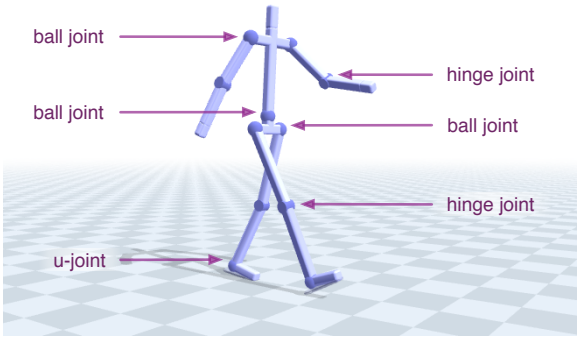

Fig. 4: Character model. *Our character has 29 degrees of freedom and consists of rigid body limbs that approximate the human body.*

can be assumed at the beginning of a set $\mathcal{T}$, we determine an action $\boldsymbol{w} \in [0,1]^l$ that is operative during the entire traversal of $\mathcal{T}$. Unlike the low-level control problem, the set of initial states and the set of possible actions are both discrete.

We use a standard policy iteration algorithm to solve this new problem. For each set $\mathcal{T}$ of composable trajectories and each initial configuration $(\boldsymbol{x}_1, \boldsymbol{\theta}_1)$, we simulate forward using the current policy, updating the cost-to-go $V(\boldsymbol{x}_1, \boldsymbol{\theta}_1)$. Then for each $\mathcal{T}$ and $(\boldsymbol{x}_1, \boldsymbol{\theta}_1)$, we search for actions $\boldsymbol{w}$ that minimize the Bellman equation (29). We iterate over this process until convergence.

6. RESULTS

We demonstrate our control framework in a number of experiments that elucidate the concepts introduced throughout this paper. We refer the reader to the companion video that further illustrates how our controllers operate.

6.1 Implementation

Our bipedal model is an articulated system with $n = 29$ degrees of freedom as shown in Figure 4. Its inertial parameters are similar to those found in earlier work [Muico et al. 2009]. Each of our components tracks a walking motion derived from motion captures recorded at 120 Hz, and each nominal trajectory is processed to compute smooth actuation torques of small magnitude while minimizing deviation from captured data. The parameters for this learning stage are listed in Appendix B.2.

We wrote an LCP-based simulation system as described in recent work [Muico et al. 2009]. We use a coefficient of restitution of 0. The friction coefficient is 0.9, but we can vary it from 0.7 to 1.6 without compromising stability. For our control system, we use the QL quadratic program solver [Schittkowski 2005].

6.2 Walking on a slope

Although a controller was designed to move on flat ground, it can also perform the same task to some degree on unlevel ground. We observed that a controller synthesized from a particular walking trajectory can successfully descend slopes as low as -9° and ascend slopes as high as 4° in simulations lasting 5 s. We attempted to improve this range by constructing controllers consisting of three

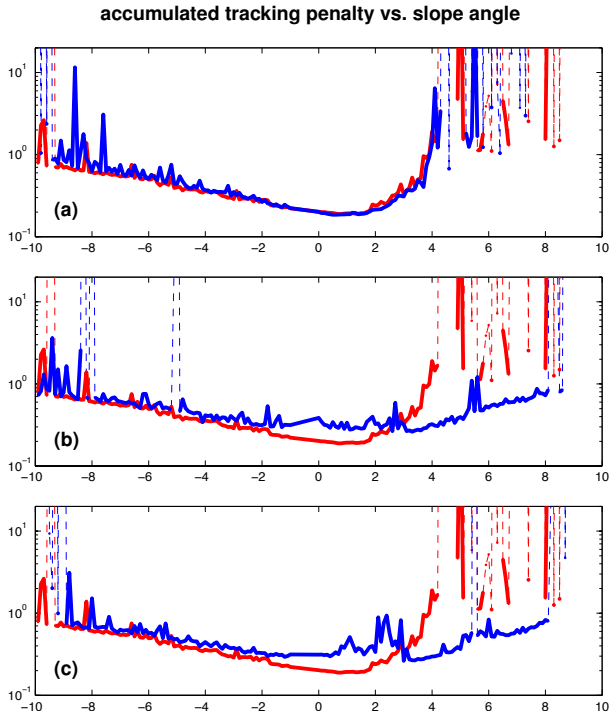

Fig. 5: Slope tests. *Three composite controllers (in blue) were repeatedly measured against the same primitive walker (in red) in simulations lasting* 5 s. *Dashed segments indicate abrupt failures.* **(a)** *A controller constructed from the original trajectory with a slightly faster variant mostly changed the downhill behavior.* **(b)** *A controller constructed from the same walking trajectory with a slightly slower variant extended the range in the uphill region but made the downhill motion slightly less stable.* **(c)** *A controller constructed from all three trajectories combined the attributes of the previous two controllers, revealing the faster trajectory in the downhill setting and the slower trajectory in the uphill setting.*

motions. Using the original trajectory, we performed spacetime optimization and generated two additional trajectories: one faster trajectory extending 0.4 m farther, and a slower trajectory receded by 0.2 m.

First we constructed a composite controller with the original trajectory and the faster trajectory and then ran 5-s simulations over a range of angles along the direction of motion. Since we use the same parameters to synthesize all our controllers, we can accumulate the state costs from Equation 11 to evaluate their relative performance. We found that the following expression is a suitable measure of fidelity

$$\frac{1}{N}\sum_{k=1}^{N}\left(c_k(\boldsymbol{x}_k, \bar{\boldsymbol{w}}_k) + \frac{1}{2}\boldsymbol{u}_k^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u}_k\right). \tag{30}$$

For the controller with the longer stride, we found that the performance remained largely unchanged as indicated in Figure 5(a).

Next we did the same experiment with the slower trajectory. We found that the downhill behavior of the composite controller was, for the most part, the same as what we observed with the single-trajectory controller. On the other hand, the slower motion manifested itself as the character decreased its speed during its ascent. As shown in Figure 5(b), this allowed the controller to succeed in tests with slopes up to 8°.

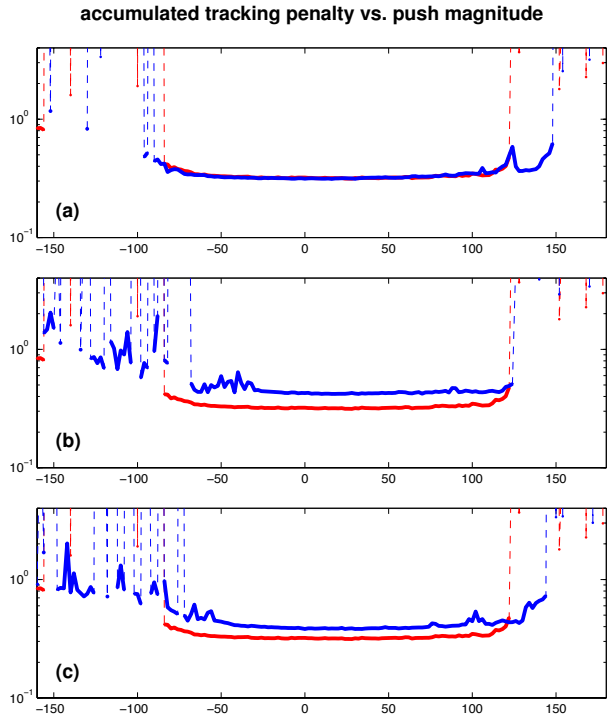

Fig. 6: Pushing tests. *Three composite controllers (in blue) were repeatedly subjected to single pushes and measured against the same primitive controller (in red) in simulations lasting* 10 s. *Dashed segments indicate abrupt failures.* **(a)** *A controller constructed from the original walking trajectory with a slightly faster variant extended the range of success in the forward direction.* **(b)** *A controller constructed from the original walking trajectory and also a different, slower trajectory mostly changed the behavior in the backward direction.* **(c)** *A controller constructed from all three trajectories combined the advantages of the previous two controllers.*

We found the same shift in its gait when we combined all three motions into one controller as indicated in Figure 5(c). We deemed this larger controller to be successful from a descent of about $-9°$ to an ascent of about $8°$.

6.3 Pushing forward

We performed another experiment on three composite controllers whose constituent motions are slightly faster than the previous ones. First, a primitive controller was synthesized from a typical walking motion, and we subjected it to an impulsive force at the torso along the direction of motion. The force was applied smoothly for a duration of about one half-cycle around the double-support phase of the gait. We observed that the operating range for this controller extends from -80 N to 120 N. Figure 6 shows the accumulated tracking costs for this test.

Like the previous slope tests, we constructed a composite controller consisting of a spacetime-optimized variant of the original walker whose stride extends 0.2 m farther. As evident in Figure 6(a), this controller outperformed the primitive controller and withstood forces up to 150 N.

Next we constructed a composite controller consisting of the original trajectory and a different slow trajectory. In contrast to the previous controller, the slower motion seems to be useful mostly for backwards pushing. While Figure 6(b) does betray the presence of some instability and inconsistency in the backwards direction, the resulting motion recovers with natural transitions, giving the appearance of the character to be significantly less stiff and more lifelike. Unfortunately, such vivid results are difficult to quantify.

Lastly, Figure 6(c) shows overall improvement in the character's robustness and clearly combines the advantages from the previous two controllers.

6.4 Pushing sideways

To demonstrate the composability of extremely disparate motions, we constructed a controller consisting of a forward walking trajectory and a few sideways trajectories. Sideways walking is an uncommon means of human locomotion, and therefore we suppress its influence in these experiments by diminishing its corresponding weights $\bar{w}^i$ by a factor of 10^{-4} relative to the dominant forward motion. That is, we expect such behavior to emerge only when the system state is compatible with sideways motion.

A key observation from the experiments is that even when a primitive controller recovers from pushes it does so in a stiff, often unnatural manner. The composite controller, in contrast, appears to exert less energy as it makes additional steps to gradually absorb the force of impact. The companion video shows that varying the impact of force creates a variety of recoveries, many of which differ from motions generated by either component. For example, different recoveries extend foot differently depending on the sideways momentum imposed by the impact; disturbances in the opposite direction produce a natural leg crossover.

6.5 Grazing and scuffing

To some extent, all our examples have utilized the constraint modification of Equation 24. In order to demonstrate its necessity in these tests, we subject a typical walking controller to various disturbances. In one particular experiment, we compared the constrained controller to an unconstrained one as they both ascended slopes of varying degree. The latter inferior controller is essentially given by Equation 9b and is equivalent to that of prior work [Muico et al. 2009]. Whereas the unconstrained controller fails on a $4°$ slope, our constrained controller succeeds on slopes twice as steep. In fact, this mechanism has played a large role in the previous slope tests involving composite controllers.

We also demonstrate the constraint's utility in pushing tests. Running motions are especially difficult to stabilize on their own, and disturbances can quickly compound such difficulties by interfering with the rythm of the gait. We subject a runner to sideways pushing at regular intervals and compared this controller to an unconstrained one. We found that the constrained controller is capable of absorbing forces up to 70 N, while the other one quickly loses balance upon such impacts.

6.6 Responding to user directives

External disturbances like those in the foregoing experiments are effective in triggering the transitions present in the motion graph, but such deliberate forces might not be typical of the character's intended operation. In our final experiment, we show that our controllers can also respond to user directives. We assembled a few

walking, turning, and sideways motions into a graph, and then we optimized over the desired orientation of the character. In other words, the user is in control of a single angle, regardless of the character's velocity. As before, we suppress the sideways motions, so that the optimization favors the more natural locomotion skills.

An interesting observation is that decisions are not based solely on the whim of the user. Rather, our control balances between what is desired by the user and what is physically possible given the current perturbation. So the character is not obligated to align with the desired orientation if the current state strongly favors one of the recovery components. Composition alleviates the many concerns about transitioning from one control to another since increased robustness and larger range of transitions increases the overall cohesion of the graph.

7. CONCLUSION

This paper presents a composable control system for three-dimensional characters. Its ability to combine the best of each component and even amplify performance of individual components has produced physically based characters that are both lifelike and robust. We contributed two methods to achieve this goal: (1) an efficient and automatic method for blending components that independently track different trajectories, and (2) a continuous-blend variation of the graph traversal strategy that allows us to transition between controllers at any point in time. Our tracking system exhibits some resilience and improves upon previous results in published literature. Interestingly, we also discovered that resulting recoveries are natural and realistic even when they are quite different from motions generated by any individual component.

Of course, this controller is only robust if it contains the right combination of components that lead to a recovery. A control system should consist of many candidate components to encourage smooth transitions. Otherwise, one can observe instability from abrupt switching between components. Currently, we assemble the appropriate trajectories by hand and explicitly specify blend weights leading to a default behavior. The problems of knowing which components are required or how to reduce the number of components are still unknown.

Another area of improvement is our high-level learning stage, a process prone to the curse of dimensionality. While we showed how we can steer the character by optimizing over a single scalar variable, we find it more difficult to scale the problem to include more task parameters. To overcome this difficulty, we believe it is worthwhile to explore optimal control problems in a reduced-dimensional setting like that used in recent work [Ye and Liu 2010].

Finally, we observed that composition does not perform as well on varying terrain, even when it performs admirably on disturbances from external pushes. Most likely, this is due to the fact that our learning algorithm does not account for ground variation. Ideally we should treat the orientation of the ground as unactuated degrees of freedom, but our control system monitors only the current state of the character. This suggests a next step for development: composite control system that react to both changes in state and the environment.

ACKNOWLEDGMENTS

The authors thank Yongjoon Lee for capturing all motion data. This work was supported by the UW Animation Research Labs, Weil Family Endowed Graduate Fellowship, UW Center for Game Science, Microsoft, Intel, Adobe, and Pixar.

APPENDIX

A. OPTIMALITY

A.1 Linear dynamics

Consider the quadratic cost-to-go function v_{k+1} in Equation 7 and a quadratic cost function

$$c_k(\boldsymbol{x}) = \frac{h}{2} \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k\right)^{\mathsf{T}} \boldsymbol{Q}_k \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k\right). \tag{31}$$

Suppose the passive dynamics at the current stage is given by

$$p_k(\cdot \mid \boldsymbol{x}) = \mathcal{N}\big(\bar{\boldsymbol{y}}_k + \boldsymbol{A}_k \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k\right), h\boldsymbol{\Sigma}_k\big). \tag{32}$$

Then v_k statisfies the Bellman equation

$$v_k(\boldsymbol{x}) = c_k(\boldsymbol{x}) + \min_{\pi} \left\{ \mathrm{KL}\left[\pi \parallel p_k\right] + \mathrm{E}_{\boldsymbol{y}\sim\pi}[v_{k+1}(\boldsymbol{y})] \right\} \tag{33}$$

if and only if v_k is the quadratic function

$$v_k(\boldsymbol{x}) = \bar{v}_k + \boldsymbol{g}_k^{\mathsf{T}} \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k\right) + \frac{1}{2} \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k\right)^{\mathsf{T}} \boldsymbol{H}_k \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k\right)$$

with coefficients given by

$$\begin{aligned} \bar{v}_k = {} & v_{k+1}(\bar{\boldsymbol{x}}_k + h\boldsymbol{a}(\bar{\boldsymbol{x}}_k)) - \frac{h}{2} \boldsymbol{u}_k(\bar{\boldsymbol{x}}_k)^{\mathsf{T}} \boldsymbol{R}_k(\bar{\boldsymbol{x}}_k) \boldsymbol{u}_k(\bar{\boldsymbol{x}}_k) \\ & + \frac{1}{2} \log\det\left(\boldsymbol{R}^{-1} \boldsymbol{R}_k(\bar{\boldsymbol{x}}_k)\right), \end{aligned} \tag{34a}$$

$$\begin{aligned} \boldsymbol{g}_k = {} & \boldsymbol{A}_k^{\mathsf{T}} \left(\boldsymbol{H}_{k+1}^{-1} + h\boldsymbol{\Sigma}_k\right)^{-1} \times \\ & \left(\bar{\boldsymbol{y}}_k - \bar{\boldsymbol{x}}_{k+1} + \boldsymbol{H}_{k+1}^{-1} \boldsymbol{g}_{k+1}\right), \end{aligned} \tag{34b}$$

$$\boldsymbol{H}_k = h\boldsymbol{Q}_k + \boldsymbol{A}_k^{\mathsf{T}} \left(\boldsymbol{H}_{k+1}^{-1} + h\boldsymbol{\Sigma}_k\right)^{-1} \boldsymbol{A}_k. \tag{34c}$$

A.2 Composite objective

Consider the cost function v_{k+1} given by Equation 10, and suppose that the following Bellman equation holds

$$\begin{aligned} v_k(\boldsymbol{x}, \boldsymbol{w}) = {} & c_k(\boldsymbol{x}, \boldsymbol{w}) \\ & + \min_{\pi} \left\{ \mathrm{KL}\left[\pi \parallel p\right] + \mathrm{E}_{\boldsymbol{y}\sim\pi}\left[v_{k+1}(\boldsymbol{y}, \boldsymbol{w})\right] \right\}. \end{aligned}$$

From the linearity of the Bellman equation, it follows that

$$\begin{aligned} c_k(\boldsymbol{x},\boldsymbol{w}) &- v_k(\boldsymbol{x},\boldsymbol{w}) \\ &= \log \int_{\mathcal{X}} p(\,\boldsymbol{y} \mid \boldsymbol{x}\,)\, e^{-v_{k+1}(\boldsymbol{y},\boldsymbol{w})}\, \mathrm{d}\boldsymbol{y} \\ &= \log \sum_{i=1}^{m} w^i e^{c_k^i(\boldsymbol{x}) - v_k^i(\boldsymbol{x})} \\ &= \log \frac{\sum_{i=1}^{m} w^i e^{c_k^i(\boldsymbol{x}) - v_k^i(\boldsymbol{x})}}{\sum_{j=1}^{m} w^j e^{-v_k^j(\boldsymbol{x})}} + \log \sum_{j=1}^{m} w^j e^{-v_k^j(\boldsymbol{x})} \\ &= \log \sum_{i=1}^{m} \alpha_k^i(\boldsymbol{x},\boldsymbol{w})\, e^{c_k^i(\boldsymbol{x})} + \log \sum_{j=1}^{m} w^j e^{-v_k^j(\boldsymbol{x})}. \end{aligned}$$

Thus $v_k(\boldsymbol{x},\boldsymbol{w}) = -\log \sum_{i=1}^{m} w^i e^{-v_k^i(\boldsymbol{x})}$ if and only if c_k satisfies

$$c_k(\boldsymbol{x},\boldsymbol{w}) = \log \sum_{i=1}^{m} \alpha_k^i(\boldsymbol{x},\boldsymbol{w})\, e^{c_k^i(\boldsymbol{x})}, \tag{35}$$

where α_k^i is given by Equation 12.

B. IMPLEMENTATION NOTES

B.1 Coordinate transformations

The value functions for each component are usually defined in different frames of reference, so we must transform them in order to solve the composite problem. Consider a transformation $\boldsymbol{f} : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ that maps a state $\boldsymbol{\xi}$ in a component reference frame to its corresponding state $\boldsymbol{x}$ in the global reference frame. We consider mappings consisting of translations, rotations and reflections. For the cost function in the component frame,

$$v(\boldsymbol{\xi}) = \bar{v} + \boldsymbol{g}^{\mathsf{T}}(\boldsymbol{\xi} - \bar{\boldsymbol{\xi}}) + \frac{1}{2}\,(\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})^{\mathsf{T}} \boldsymbol{H}\,(\boldsymbol{\xi} - \bar{\boldsymbol{\xi}}),$$

the corresponding cost function in the global frame is given by

$$\begin{aligned} v(\boldsymbol{x}) = \bar{v} &+ \left(\mathrm{D}\boldsymbol{f}(\bar{\boldsymbol{\xi}})\,\boldsymbol{g}\right)^{\mathsf{T}} \left(\boldsymbol{x} - \boldsymbol{f}(\bar{\boldsymbol{\xi}})\right) \\ &+ \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{f}(\bar{\boldsymbol{\xi}})\right)^{\mathsf{T}} \mathrm{D}\boldsymbol{f}(\bar{\boldsymbol{\xi}})\, \boldsymbol{H}\, \mathrm{D}\boldsymbol{f}(\bar{\boldsymbol{\xi}})^{\mathsf{T}} \left(\boldsymbol{x} - \boldsymbol{f}(\bar{\boldsymbol{\xi}})\right). \end{aligned} \tag{36}$$

B.2 Learning parameters

Table I. : *Objective weights*

feature	DOF	$\boldsymbol{Q}_{\mathrm{coord}}$	$\boldsymbol{Q}_{\mathrm{vel}}$	$\boldsymbol{R}\,(\times 10^{-5})$
height	1	10	0.32	–
location	2	4.0	0.30	–
orientation	3	12	0.22	–
trunk joint	3	12	0.18	0.40
ankle joint	2	5.0	0.20	0.30
knee joint	1	8.0	0.12	0.30
hip joint	3	12	0.15	0.30
shoulder joint	3	6.0	0.075	0.40
elbow joint	1	4.0	0.050	0.50
heel position	3	–	–	0.040
toe position	3	–	–	0.040

REFERENCES

BURRIDGE, R. R., RIZZI, A. A., AND KODITSCHEK, D. E. 1999. Sequential composition of dynamically desterous robot behaviours. *International Journal of Robotics Research 18,* 6, 534–555.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2009. Robust Task-based Control Policies for Physics-based Characters. *ACM Transactions on Graphics (SIGGRAPH Asia)*.

COROS, S., BEAUDOIN, P., YIN, K., AND VAN DE PANNE, M. 2008. Synthesis of constrained walking skills. *ACM Transactions on Graphics 27,* 5, 113:1–113:9.

DA SILVA, M., ABE, Y., AND POPOVIĆ, J. 2008. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27,* 3, 82:1–82:10.

DA SILVA, M., DURAND, F., AND POPOVIĆ, J. 2009. Linear bellman combination for control of character animation. *ACM Transactions on Graphics 28,* 3, 82:1–82:10.

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics 29,* 3.

EREZ, T. AND SMART, W. 2007. Bipedal walking on rough terrain using manifold control. *International Conference on Intelligent Robots and Systems (IROS)*, 1539–1544.

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.

FLEMING, W. H. 1978. Exit probabilities and optimal stochastic control. *Applied Mathematics and Optimization 4*, 329–346.

HODGINS, J. K. AND POLLARD, N. S. 1997. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH 97*. 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.

HOLLAND, C. 1977. A new energy characterization of the smallest eigenvalue to the schrödinger equation. *Communications on Pure and Applied Mathematics 30*, 755–765.

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, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Transactions on Graphics 29,* 4.

MORDATCH, I., DE LASA, M., AND HERTZMANN, A. 2010. Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics 29,* 3.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28,* 3, 81:1–81:9.

POLLARD, N. S. AND BEHMARAM-MOSAVAT, F. 2000. Force-based motion editing for locomotion tasks. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 663–669.

RAIBERT, M., BLANKESPOOR, K., NELSON, G., AND PLAYTER, R. 2008. Bigdog, the rough-terrain quadruped robot. In *Proceedings of International Federation of Automatic Control*.

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

SCHITTKOWSKI, K. 2005. *QL: A Fortran Code for Convex Quadratic Programming – User's Guide, Version 2.11*. Department of Mathematics, University of Bayreuth.

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

TODOROV, E. 2009a. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems (NIPS)*. Vol. 22. 1856–1864.

TODOROV, E. 2009b. Efficient computation of optimal actions. In *Proceedings of the National Academy of Sciences (PNAS)*. Vol. 106. 11478–11483.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2010. Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics 29,* 4.

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

YE, Y. AND LIU, C. K. 2010. Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics 29,* 4.

YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2008. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics 27,* 3, 81:1–81:7.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: simple biped locomotion control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. 105.

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