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

Physically Plausible Simulation for Character Animation

Sergey Levine[†1] and Jovan Popović[2]

[1] Stanford University, Department of Computer Science
[2] Adobe Systems Inc.

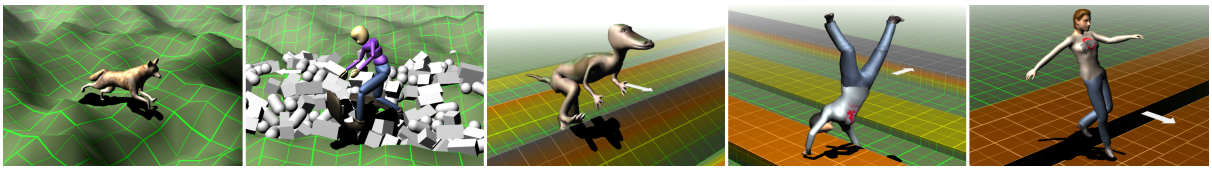

Figure 1: *Our method adapts keyframed and motion captured characters to rough terrain, simulated rubble, and steep slopes, allows a cartwheeling character to traverse high steps, and causes a walking character to balance on shifting platforms.*

Abstract

Artist-created animated characters can exhibit stylized, engaging behavior, but require considerable effort to construct, while interactive applications require numerous motions and variations to create a dynamic, believable character. This paper describes a method for generating some of these variations automatically: given a stream of poses, our method simulates plausible responses to physical disturbances and environmental variations. Our quasi-physical simulation accounts for the dynamics of the character and surrounding objects, but does not require the motion to be physically valid, making it suitable for both realistic and stylized, cartoony motions. It further does not require any preprocessing, allowing it to run as an online filter that transforms the output of any real-time animation system. Our prototype runs at 50 Hz, on bipeds and quadrupeds with over 50 degrees of freedom, and generates plausible variations for walking, running, hopping, crawling, rolling, cartwheeling, and other motions.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Games and virtual worlds require engaging animated characters. Many of the most compelling, engaging, and endearing characters are hand-animated by professional artists, who imbue their motions with rich, expressive personality without the constraints of physical realism. However, dynamic surroundings and physical interactions can create scenarios where premade motions are no longer suitable: a character might be pushed by a large force, or, as shown in Figure 1, might be required to walk on simulated rubble. A motion library consisting entirely of compelling, hand-created motions that covers the space of all possible reactions to such scenarios would be prohibitively time consuming to create.

Physical simulation offers the promise of adapting virtual characters to an infinite variety of situations by using their equations of motion to predict plausible reactions. However, this requires controllers that can reproduce the desired motions, since kinematic animations alone do not provide the necessary feedback laws to simulate the motion and stabilize it in the presence of disturbances. Constructing robust control schemes for arbitrary activities remains an open problem. Furthermore, stylized cartoony motions, which are not physically valid, cannot be reproduced in a physical simulation with any set of torques. To handle such motions, we require simulations that are physically *plausible*, but not necessarily physically valid, so that any input motion can be simulated while retaining its original character [BHW96].

† e-mail: svlevine@cs.stanford.edu

© The Eurographics Association 2012.

In this paper, we propose a quasi-physical simulation method that enriches the motions of virtual characters with plausible responses and adaptations. In the absence of perturbations, the input motion remains unchanged, even if it is stylized and non-physical. In the presence of disturbances, the simulation resolves unexpected contacts, adjusts foot placement, and deflects joints affected by external forces and collisions. These adjustments make use of the character's equations of motion, but are allowed to violate physics in order to follow non-physical reference animations and prevent undesired failure, such as falls. This allows our approach to preserve the potentially non-physical flavor of the original motion while still simulating appropriate responses.

All processing is performed online as the reference frames are supplied by a motion clip or kinematic controller. This allows our method to be seamlessly integrated into existing animation pipelines, and makes it compatible with all prior kinematic controllers, including those that generate motions dynamically or without regard for physical validity. Such kinematic controllers could leverage graphs and optimal control [TLP07], move trees [Joh09], or even human push response demonstrations [AFO05]. In our prototype, we use a simple kinematic controller that responds to changes in the character's velocity. When combined with such controllers, the quasi-physical simulation produces "unconscious" physical responses and simple adaptations, while the controller is responsible for any "conscious" or "intelligent" responses. In this way, we can leverage the substantial existing literature on kinematic control to produce characters that appear both intelligent and physically plausible.

Relaxing the equations of motion also allows us to enforce goal constraints that ensure that the character moves in the desired direction and does not fall, regardless of how unstable or unbalanced the reference motion might be. This provides the content creator the freedom to use any reference motion or kinematic controller, without the need to author additional logic to handle failure cases. The constraint can be deactivated to allow the character to lose balance, but the crucial difference is that the content author has control over the character's reliability. Such guarantees of reliability are required for many interactive applications, but are generally lacking in fully physical control schemes.

We also integrate our proposed quasi-physical simulation directly into an existing rigid body simulation engine to enable two-way interactions between the character and simulated objects, while still following non-physical, stylized motions perfectly in the absence of perturbations. Preserving the character of the original motion while enabling two-way coupling has proven exceptionally challenging for prior methods that utilize non-physical forces, since it requires specifying the forces directly for use in a fully physical simulation [WJM06]. We address this problem by directly incorporating the quasi-physical objective into the equations of motion of the character. The new equations of motion maximize the quasi-physical objective implicitly as part of the rigid body simulation, and remove the need to construct and tune feeedback policies such as PD servos.

The main contribution of this paper is a real-time method for augmenting any kinematic animation with physically-based responses to external perturbations and variation in the environment. Our method does not enforce or require physical consistency on either the input or output motion, making it suitable even for cartoony, artist-animated characters. Using a novel reformulation of the equations of motion, our method can be integrated with existing rigid body simulators to simulate quasi-physical characters that interact with simulated dynamic objects. Because our quasi-physical simulation is an online post-process compatible with any kinematic animation method, the reference motions need not be available in advance for pre-processing. Since our method can follow even non-physical motions, the style of the reference motion is preserved. Furthermore, because the method admits goal constraints, the designer need not handle failure cases, such as standing up after a fall, unless a fall is intentionally allowed to occur.

1.1. Related Work

Previous methods for animating responses to perturbations and environmental variation can be broadly categorized as either fully physical, fully kinematic, or hybrid. Considerable recent progress has been made on fully physical methods that can automatically simulate specific behaviors without input animation data [YLvdP07, WP10, WFH10, MdLH10, dLMH10]. While such methods can produce good results, they limit the degree of stylistic expression to those motions that can be produced by the controller. In this work, we instead address the problem of adding physical responses to characters that exhibit an artist-specified behavior, rather than creating the behavior from scratch. A number of physical methods follow motion capture data within a physical simulation. However, most such methods often require the motion to be known in advance to construct a control policy [SvdP05, dSAP08], or to make the reference motion physically consistent [SKL07, LYvdP*10]. Those that do not require preprocessing either deal with statically balanced tasks [MZS09], or use task-specific control or balance laws [LKL10].

Fully physical methods are always susceptible to failure under large disturbances, which our method avoids by using non-physical forces. In many applications, even physically correct but undesired failures can frustrate the user and place undue burden on the content creator, who must prevent conditions that lead to failure, or create additional logic to handle them. Although simulation is often used in modern games for animating unactuated (e.g. dead or unconscious) characters, these limitations have slowed broader adaption of physically valid methods for character animation. In addition, although our approach sacrifices physical validity, it

© The Eurographics Association 2012.

can operate directly on non-physical kinematic motion data, without any preprocessing. Our method can also handle a much broader range of input motions than fully physical methods, since the use of non-physical forces allows us to follow even stylized "cartoon" motions. No fully physical method can possibly follow a reference motion that is itself inherently nonphysical.

Kinematic methods reassemble and procedurally modify example animations to accomplish a desired task. However, such methods are limited in their ability to respond to perturbations. Kinematic methods that can synthesize perturbation responses generally require a substantial number of example clips showing good recoveries [AFO05, YL10]. In contrast, our method can produce plausible responses from even a single unperturbed example clip by using the character's equations of motion.

Finally, several methods have been developed that attempt to find a compromise between accurate simulation and kinematic control. Such methods selectively apply simulation either to a subset of the character's joints [OTH02, YL08, vWZR09], or during specific time intervals [ZMcCF05]. Since the character is partially controlled in a fully kinematic manner, such methods necessarily limit the range of possible responses. For example, methods that kinematically animate the lower body cannot exhibit lower body responses to unexpected physical contacts. On the other hand, methods that switch to fully physical control in response to perturbations still require physically valid reference motions and elaborate feedback policies. With our method, the entire character is aware of physically perturbations at all times, enabling a range of responses that is comparable to fully physical methods, while the non-physical, stylized aesthetic of the reference motions is faithfully preserved.

To follow non-physical reference motions and enforce goal constraints, our technique applies minimal forces to the unactuated root of the character. The use of subtle non-physical forces has been proposed as a reasonable model for producing visually plausible animations when some parameters of the physical model are not fully known [BHW96]. Our work strives to bring this idea to character animation, in order to combine the generality of physics with the reliability and flexibility of kinematic techniques. Minimization of root forces has previously been applied as an intermediate stage in controller training [VL95], and several prior methods have proposed applying forces to the root to simplify control [WJM06, Hor08]. However, without explicit minimization of these forces, the character can appear "marionette-like" and non-physical even under small perturbations. Furthermore, such previous methods still supply torques to a conventional rigid body simulator. This requires carefully tuned feedback policies, and the reference motions are usually not reproduced accurately, even in the absence of perturbations.

1.2. Overview

The quasi-physical simulation accepts as input the previous, current, and next desired pose for the character, as well as the environment and any external forces, and computes the next pose for the character by solving an optimization problem that adapts the character to its surroundings. The desired poses may be taken directly from an animation clip, or produced dynamically by a kinematic controller. The simulated pose exhibits physics-based responses to pushes and pulls, adjusts the feet to walk on terrains of varying height, and interacts with other simulated objects. The motion generated by the quasi-physical simulation also obeys a goal constraint. In our prototype, this constraint forces the character's center of mass to follow a desired trajectory. Games often specify the path of either the root or the center of mass directly, using an animation module to generate full-body animations for the desired trajectory [Joh09]. This makes center of mass motion a natural goal constraint, although other goal constraints are also supported.

The quasi-physical simulation is able to follow non-physical reference motions and always satisfy the goal constraint by applying non-physical forces on the character's root. To ensure that motions remain plausible, the simulation minimizes the difference between the current root forces and those observed in the reference motion. This means that if the reference motion is physical, the root forces are minimized, but if it is non-physical, quasi-physics can still track the motion accurately. The result is that quasi-physics produces physically plausible behavior when disturbances are small, and ensures that the goal constraints are satisfied when they are large.

In a static environment, the quasi-physical simulation can be performed directly by solving a quadratic program. However, if a rigid body simulator is already used to animate dynamic objects, we can integrate quasi-physical simulation into it directly by including the quasi-physical objective in the character's equations of motion. This allows us to use an existing rigid body contact solver to enable two-way coupling between the character and simulated objects, while still allowing the character to follow the reference motion and maintain the guarantees afforded by the goal constraint. Unlike traditional physical animation methods, this approach does not require us to construct a feedback policy that computes appropriate joint torques. Instead, the optimal torques are produced automatically by the existing contact solver.

2. Quasi-Physical Control

We call our approach "quasi-physical," because it uses the character's equations of motion, but permits small non-physical forces to be applied at the character's root in order to follow non-physical reference motions and prevent undesired motions (such as falling). Actuation of the root is essential for tracking non-physical motions, but excessive root

© The Eurographics Association 2012.

actuation can result in visual artifacts, making the character look like a marionette held up by invisible strings. The key to our approach is the explicit minimization of unnecessary root actuation, which makes the non-physical forces unnoticable to a casual observer unless very large perturbations are applied. Since the soft root unactuation makes any acceleration feasible, we can also impose hard goal constraints to guide the character. Quasi-physical control takes the form of a constrained optimization that computes compatible joint accelerations $\boldsymbol{a}_k$ and joint torques $\boldsymbol{\tau}_k$:

$$(\boldsymbol{a}_k, \boldsymbol{\tau}_k) = \arg\min_{\boldsymbol{a}_k, \boldsymbol{\tau}_k} E(\boldsymbol{\tau}_k, \boldsymbol{a}_k) \quad (1)$$

$$\text{s.t.}\, \boldsymbol{G}(\boldsymbol{a}_k) = 0 \quad (2)$$

$$\boldsymbol{D}(\boldsymbol{a}_k) = \boldsymbol{\tau}_k. \quad (3)$$

Equations 2 and 3 enforce the goal constraints and equations of motion. The objective is defined as the sum of three quadratic terms: the torque term $E_\tau(\boldsymbol{\tau}_k)$, the pose term $E_\mathrm{p}(\boldsymbol{a}_k)$, and the end-effector term $E_\mathrm{e}(\boldsymbol{a}_k)$. Intuitively, these terms can be thought of as enforcing physicality, tracking the reference motion, and adapting to the environment, respectively.

2.1. Equations of Motion

We represent a character as a kinematic tree consisting of rigid links and revolute joints. The root of the kinematic tree, conventionally taken to be the pelvis, is "attached" to the world with a six degree of freedom joint. When this joint is unactuated, the kinematic tree is said to have a floating base [Fea07]. The pose of the character at time step k may be described by a vector $\boldsymbol{q}_k$, each entry of which is the current angle of a joint, with the first three entries giving the position of the root in Cartesian coordinates.

The velocities of the character's joints are computed with finite differences, according to

$$\boldsymbol{v}_k = \frac{\boldsymbol{q}_k - \boldsymbol{q}_{k-1}}{\Delta t}. \quad (4)$$

The accelerations of the joints are related to the current, previous, and next pose according to

$$\boldsymbol{a}_k = \frac{\boldsymbol{q}_{k+1} - 2\boldsymbol{q}_k + \boldsymbol{q}_{k-1}}{\Delta t^2}. \quad (5)$$

These finite difference equations correspond to a semi-implicit integration step, which is often used in real time simulators:

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \Delta t \boldsymbol{a}_k$$
$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \Delta t \boldsymbol{v}_{k+1}.$$

To be physically consistent, the character's accelerations $\boldsymbol{a}_k$ and torques $\boldsymbol{\tau}_k$ must respect the equations of motion $\boldsymbol{D}(\boldsymbol{a}_k) = \boldsymbol{\tau}_k$:

$$\boldsymbol{M}(\boldsymbol{q}_k)\boldsymbol{a}_k + \boldsymbol{h}(\boldsymbol{q}_k, \boldsymbol{v}_k) = \boldsymbol{J}_\mathrm{e}^T \boldsymbol{f}_\mathrm{e} + \boldsymbol{J}_\mathrm{c}^T \boldsymbol{f}_\mathrm{c} + \boldsymbol{\tau}_k, \quad (6)$$

where $\boldsymbol{M}(\boldsymbol{q}_k)$ is the pose-dependent mass matrix, which can be computed with the Composite Rigid Body algorithm [Fea07], the inertial term $\boldsymbol{h}(\boldsymbol{q}_k, \boldsymbol{v}_k)$ accounts for Coriolis, centripetal, and gravitational forces, computed with Recursive Newton-Euler [Fea07], $\boldsymbol{f}_\mathrm{c}$ represents contact forces, and $\boldsymbol{f}_\mathrm{e}$ represents perturbation forces, such as pushes and pulls. The Jacobians $\boldsymbol{J}_\mathrm{c}$ and $\boldsymbol{J}_\mathrm{e}$ transform the forces into body coordinates. The masses of the individual links are estimated, for simplicity, to be proportional to their lengths, though the user can specify link masses directly.

In a rigid body simulation, the equations of motion can be solved for $\boldsymbol{a}_k$ when $\boldsymbol{\tau}_k$ is known [Fea07], while many control methods optimize an objective in terms of $\boldsymbol{\tau}_k$, subject to the consraints imposed by the equations of motion. Since the character's root is unactuated, physically consistent controllers also fix the first six entries of $\boldsymbol{\tau}_k$ to zero. Our formulation instead strives minimize the impact of these forces without eliminating them entirely.

2.2. Goal Constraint

The goal constraint offers a general method for guiding the character. Its particular form will often depend on the application, and many variations are possible. Our prototype guides the character by controlling the horizontal trajectory of its center of mass. This constraint ensures that the horizontal position of the center of mass at the next time step, denoted $\boldsymbol{o}_{k+1}$, matches the desired position $\boldsymbol{o}^\star_{k+1}$. To express the constraint in terms of joint accelerations, we use Equation 5 to expand $\boldsymbol{o}_{k+1}$ and rewrite its acceleration using the center of mass Jacobian $\boldsymbol{J}_\mathrm{c}$:

$$\boldsymbol{G}(\boldsymbol{a}_k) = 2\boldsymbol{o}_k - \boldsymbol{o}_{k-1} + \Delta t^2 (\boldsymbol{J}_\mathrm{c}\boldsymbol{a}_k + \dot{\boldsymbol{J}}_\mathrm{c}\boldsymbol{v}_k) - \boldsymbol{o}^\star_{k+1} = 0.$$

The Jacobian $\boldsymbol{J}_\mathrm{c}$ is computed as the mass-weighted sum of the Jacobians of the joints, and $\dot{\boldsymbol{J}}_\mathrm{c}\boldsymbol{v}_k$ is the mass-weighted sum of joint accelerations under zero actuation, which is obtained as a by-product of the Recursive Newton-Euler algorithm and sometimes referred to as velocity-product acceleration [Fea07]. This is just one of many possible goal constraint choices, and other constraints can be formulated in a similar fashion, by expressing the constraint as a function of joint angles and differentiating to derive a linear equation.

2.3. Torque Objective

The torque objective term $E_\tau(\boldsymbol{\tau}_k)$ penalizes both root actuation and large joint forces. Root forces should be penalized to avoid non-physically actuating the root, which can give the character the appearance of a marionette. Gently penalizing the other torques helps the character appear compliant under perturbations. The torque term minimizes the difference between the current torques and a vector $\boldsymbol{\tau}^\star_k$ of desired joint torques, according to a scaling matrix $\boldsymbol{W}_\tau$ that corresponds to the inverse "strength" of each joint:

$$E_\tau(\boldsymbol{\tau}_k) = \|\boldsymbol{\tau}_k - \boldsymbol{\tau}^\star_k\|^2_{\boldsymbol{W}_\tau}.$$

© The Eurographics Association 2012.

For non-root joints i, we found that setting $\boldsymbol{W}_\tau(i)$ to the inverse of the mass that is rooted at that joint generally produced good results. For example, the stance foot would have a weight equal to the inverse of the total mass m, since it must support the whole body, while the weight on the wrist would be the inverse of the hand's mass. The user can of course modify these weights to favor some joints over others. For example, we found it useful to increase the strength of spine joints by a factor of 2 to 4, since the trunk is stronger than limbs of similar length.

To favor actuated joints over non-actuated root joints, the root joints are weighted higher. In our implementation, we set $\boldsymbol{W}_\tau(i_u) = 500/m$ for root joints $i_u = \{1,..,6\}$. We found this value to work well for both realistic, motion-captured reference motions, and for non-realistic, stylized animations. Higher weights increase realism but decrease the fidelity with which the reference motion can be followed, while lower weights create faster disturbance rejection at the cost of stiffness and lower realism.

Setting $\boldsymbol{\tau}_k^\star$ to 0 will minimize all joint forces. However, non-physical reference motions contain intentional actuation of the root, and removing such actuation can obliterate non-physical detail. We therefore estimate the torques needed to follow the reference motion in the absence of perturbations by solving a small optimization for $\boldsymbol{\tau}_k^\star$, using the reference pose, velocities, and accelerations $\boldsymbol{q}_k^\star$, $\boldsymbol{v}_k^\star$, and $\boldsymbol{a}_k^\star$, which are obtained with Equations 4 and 5:

$$\begin{aligned} &\min_{\boldsymbol{\tau}_k^\star, \boldsymbol{\lambda}} \|\boldsymbol{\tau}_{k1,..,6}^\star\|^2 \\ &s.t. \boldsymbol{M}(\boldsymbol{q}_k^\star)\boldsymbol{a}_k^\star + \boldsymbol{h}(\boldsymbol{q}_k^\star, \boldsymbol{v}_k^\star) - \boldsymbol{J}_c^T \boldsymbol{V}_c \boldsymbol{\lambda} = \boldsymbol{\tau}_k^\star \\ &\quad \boldsymbol{\lambda} \geq 0. \end{aligned}$$

the first six entries of $\boldsymbol{\tau}_k^\star$, denoted $\boldsymbol{\tau}_{k1,..,6}^\star$, are the root torques. The contact forces are represented by the polygonal friction cone $\boldsymbol{V}_c\boldsymbol{\lambda}$, which is discussed in more detail in Section 3. Although contacts exhibit nonlinear complementarity constraints, these constraints do not appear in this optimization since the accelerations are fixed. Instead, we need only replace $\boldsymbol{V}_c$ with a single basis vector in the case of sliding contacts, or, as in our prototype, simply exclude such contacts from consideration under the assumption that only fixed contacts exert significant force in the reference motion. Since only the first six entries of $\boldsymbol{\tau}_k^\star$ appear in the objective, the remaining entries can be excluded from the optimization and recovered in closed form once $\boldsymbol{\lambda}$ is computed, resulting in a very small and fast quadratic program. With correct reference torques, the quasi-physical control objective can perfectly follow any reference motion in the absence of perturbations, which means that the joint strengths $\boldsymbol{W}_\tau$ provide an intuitive knob for tweaking the stiffness of perturbation responses without affecting unperturbed motion.

2.4. Pose Objective

The pose objective term $E_p(\boldsymbol{a}_k)$ tracks a reference motion generated by a kinematic controller. This term only requires the k^{th} pose, the previous pose, and the next pose. To track the reference motion, we minimize the distance between the current acceleration $\boldsymbol{a}_k$ and a target acceleration computed according to a PD control rule. Note that PD control is applied to accelerations rather than forces. The objective term, which minimizes the difference between $\boldsymbol{a}_k$ and the PD acceleration, is given by

$$E_p(\boldsymbol{a}_k) = \|\boldsymbol{a}_k - \boldsymbol{a}_k^\star - (\boldsymbol{v}_k^\star - \boldsymbol{v}_k)\omega_p\zeta_p - (\boldsymbol{q}_k^\star - \boldsymbol{q}_k)\omega_p^2\|_{\boldsymbol{W}_p}^2.$$

where ω_p and ζ_p determine PD gains, and $\boldsymbol{W}_p$ is a scaling matrix consisting of the fraction of the character's mass rooted at each joint. $\boldsymbol{q}_k^\star$, $\boldsymbol{v}_k^\star$, and $\boldsymbol{a}_k^\star$ are the reference pose, velocity, and acceleration at the current time step. We set the ratio to critical damping, with $\zeta_p = 1$. The frequency ω_p controls how quickly the character returns to the reference motion. We use $\omega_p = 20$ in all examples, though we observed that a wide range of values produced reasonable results. We also weigh the objective against the torque term by multiplying all weights by a constant, 2.5 in all examples.

We can obtain the reference pose, velocity, and acceleration from the current, next, and previous reference poses directly by using the finite difference equations 4 and 5. However, we can obtain better results by tracking the joint orientations and velocities in world-space rather than parent-space, as observed by Wrotek et al. [WJM06]. When tracking in body coordinates, errors from parent joints propagate to their children, resulting in artifacts when joints deep in the kinematic tree (such as the pelvis) deviate from their targets. To track a joint i in world-space, we compute a parent-space orientation $\boldsymbol{q}_k^\star(i)$ that causes the joint's world-space orientation to match the k^{th} reference pose as closely as possible, given the current orientation of its parent. We denote the world-space rotation matrix of the parent of joint i with ${}_0^{p_i}\boldsymbol{T}_k$, the parent-space rotation of i with ${}_{p_i}^{i}\boldsymbol{T}_k$, and rotations in the reference poses with ${}_0^{p_i}\boldsymbol{T}_k^\star$ and ${}_{p_i}^{i}\boldsymbol{T}_k^\star$. The reference world-space orientation of joint i at frame k is given by ${}_0^{p_i}\boldsymbol{T}_k^\star\,{}_{p_i}^{i}\boldsymbol{T}_k^\star$, and we can convert it to parent-space in the current pose by multiplying it by ${}_0^{p_i}\boldsymbol{T}_k^{-1}$. We could obtain the targets for steps $k-1$ and $k+1$ in the same way, but ${}_0^{p_i}\boldsymbol{T}_{k+1}^{-1}$ is not available. Instead, we use the parent orientations at step k throughout to obtain the following desired rotations:

$$\begin{aligned} \boldsymbol{R}_{k-1}^\star(i) &= {}_0^{p_i}\boldsymbol{T}_k^{-1}\,{}_0^{p_i}\boldsymbol{T}_k^\star\,{}_{p_i}^{i}\boldsymbol{T}_{k-1}^\star \\ \boldsymbol{R}_k^\star(i) &= {}_0^{p_i}\boldsymbol{T}_k^{-1}\,{}_0^{p_i}\boldsymbol{T}_k^\star\,{}_{p_i}^{i}\boldsymbol{T}_k^\star \\ \boldsymbol{R}_{k+1}^\star(i) &= {}_0^{p_i}\boldsymbol{T}_k^{-1}\,{}_0^{p_i}\boldsymbol{T}_k^\star\,{}_{p_i}^{i}\boldsymbol{T}_{k+1}^\star. \end{aligned}$$

From $\boldsymbol{R}_k^\star(i)$, we compute the desired joint orientation $\boldsymbol{q}_k^\star(i)$ by converting $\boldsymbol{R}_k^\star(i)$ into exponential coordinates (also known as axis-angle) and projecting it onto the axis of joint i. The target orientation is then clamped to be consistent with joint limits.

© The Eurographics Association 2012.

2.5. End-Effector Objective

The end-effector objective term $E_e(\boldsymbol{a}_k)$ handles joints that are currently in contact with the environment, as well as joints that must soon be in contact with the environment according to the reference motion. Since the character can only exert external forces by contacting the environment, special handling of contacts ensures that they are established at the right time and produce a physically plausible animation. We formulate the end-effector objective as a PD controller on task-space (Cartesian) end effector accelerations, using the end-effector Jacobian $\boldsymbol{J}_e$ to transform accelerations from body coordinates to task-space:

$$E_e(\boldsymbol{a}_k) = \\ \|\boldsymbol{J}_e\boldsymbol{a}_k + \dot{\boldsymbol{J}}_e\boldsymbol{v}_k - \boldsymbol{a}_{ek}^\star - (\boldsymbol{v}_{ek}^\star - \boldsymbol{J}_e\boldsymbol{v}_k)\omega_e\zeta_e - (\boldsymbol{p}_{ek}^\star - \boldsymbol{p}_{ek})\omega_e^2\|_{\boldsymbol{W}_e}^2$$

We use $\boldsymbol{p}_{ek}$, $\boldsymbol{v}_{ek}$, and $\boldsymbol{a}_{ek}$ to denote task-space positions, velocities, and accelerations of end-effectors, and $\boldsymbol{p}_{ek}^\star$, $\boldsymbol{v}_{ek}^\star$, and $\boldsymbol{a}_{ek}^\star$ to denote their desired positions, velocities, and accelerations. We set ζ_e to $\frac{1}{2}$ and ω_e to 20.

Target Positions. For end-effector target positions $\boldsymbol{p}_{ek}^\star$, we can use the task-space positions of the end-effector joints in the reference motion and obtain $\boldsymbol{v}_{ek}^\star$ and $\boldsymbol{a}_{ek}^\star$ using equations 4 and 5. However, on uneven terrain, end-effector positions must be adjusted to clear obstacles and land on uneven surfaces. In our prototype, we add the height of the terrain at the end-effector's location to its height in the reference motion to obtain height-corrected targets. We further improve the end-effector target to allow it to clear steps by checking the height at the position $\boldsymbol{p}_{ek} + t_\ell \boldsymbol{v}_{ek}$ the end-effector is predicted to occupy a short amount of time t_ℓ in the future. We use $t_\ell = 0.2$, or 10 time steps. Prior work has suggested more complex methods for adjusting foot targets [Joh09, WP10], which could be substituted for our simple approach.

To prevent contacting feet from slipping excessively, we offset the targets for joints that are in contact by the difference between the actual and target positions of the joint at the time the contact occurs. This causes the target to "stick" to the position where the contact occurs, but still allows it to slip if the reference motion has slippage.

While the end-effector is in contact, we can use equations 4 and 5 to get $\boldsymbol{v}_{ek}^\star$ and $\boldsymbol{a}_{ek}^\star$. However, when it is not in contact, its height can change non-smoothly due to abrupt variations in the terrain (such as stairs). To prevent the PD controller from tracking these abrupt variations, we use $\boldsymbol{v}_{ek}^\star$ and $\boldsymbol{a}_{ek}^\star$ directly from the reference motion, without the height offset.

Weights The weight of an end-effector in the diagonal matrix $\boldsymbol{W}_e$ accounts for the time until the joint will be in contact in the reference motion and the joint's current height. The temporal weight encourages joints to contact the environment at the right time, while the height weight prevents joints that are too low to the ground from causing the character to trip, and prevents contacting joints from penetrating the environment. Since inter-penetration and foot height are only relevant in the direction normal to the supporting surface, we only apply this weight to the normal direction. When the normal is not vertical, we rotate $\boldsymbol{p}_e$, $\boldsymbol{v}_e$, and $\boldsymbol{a}_e$ so that the vertical axis is aligned with this normal and can be weighted separately.

The tangent weight is given by $1 - t_h/t_m$, where t_h is the estimated time until the end-effector in the reference motion comes into contact and t_m is the maximum time at which we begin to track the end-effector (0.3 seconds in our prototype). The normal weight is the sum of the tangent weight and $1 - h/h_m$, where h is the distance from the end effector to the environment (clamped to h_m), and h_m is the maximum height at which we begin tracking, set to 20 cm. As with the pose objective, we multiply all weights by a constant value of 25 to balance them against the torque term.

3. Quasi-Physical Simulation

Quasi-physical simulation requires minimizing the constrained control objective in Equation 1 at each frame. The objective is quadratic and the goal constraints are linear. However, the equations of motion in Equation 3 produce a nonlinear constraint, because the relation between the torques $\boldsymbol{\tau}_k$ and accelerations $\boldsymbol{a}_k$ depends on the contact forces. These forces vary nonlinearly with $\boldsymbol{a}_k$ due to complementarity conditions, which ensure, for example, that friction always opposes the direction of motion. Expanding the equations of motion, we can write Equation 1 as

$$\begin{aligned} &\min_{\boldsymbol{a}_k, \boldsymbol{\tau}_k, \boldsymbol{\lambda}} E(\boldsymbol{\tau}_k, \boldsymbol{a}_k) \\ &\quad s.t.\, \boldsymbol{G}(\boldsymbol{a}_k) = 0 \\ &\qquad \boldsymbol{M}(\boldsymbol{q}_k)\boldsymbol{a}_k + \boldsymbol{h}(\boldsymbol{q}_k, \boldsymbol{v}_k) = \boldsymbol{J}_e^T\boldsymbol{f}_e + \boldsymbol{J}_c^T\boldsymbol{f}_c + \boldsymbol{\tau}_k \\ &\qquad \boldsymbol{f}_c^i \in \boldsymbol{V}_c^i, \quad \ell(\boldsymbol{f}_c^i) = 0 \quad \forall i, \end{aligned}$$

where $\boldsymbol{f}_c^i$ is the contact force for the i^{th} contact, $\ell(\boldsymbol{f}_c^i)$ represents the complementarity conditions, and the linear constraint $\boldsymbol{f}_c^i \in \boldsymbol{V}_c^i$ ensures that the forces lie within a polygonal friction cone. Further details regarding this contact model can be found in prior work [AP97]. Prior methods that optimize such objectives for character control generally drop the nonlinear complementarity conditions $\ell(\boldsymbol{f}_c^i) = 0$, turning the optimization into a quadratic program [dSAP08, dLMH10]. We can perform the same approximation for quasi-physical simulation, which produces an efficient algorithm that requires only a QP solver. However, because the QP ignores complementarity conditions, it can produce unrealistic contacts. For example, the frictional force is allowed to accelerate the contact forward, rather than just opposing the direction of motion. Furthermore, environments with dynamic rigid bodies necessitate two-way coupling to ensure that the character can interact convincingly with simulated objects. This would require including the equations of motion of all the rigid bodies in the QP constraints, which is impractical.

© The Eurographics Association 2012.

On the other hand, existing rigid body simulators already handle complementarity conditions and provide for two-way coupling by solving a linear complementarity problem (LCP). In the following sections, we describe an alternative approach to optimizing Equation 1 that leverages such existing simulators, providing both complementarity conditions and two-way coupling. This approach involves integrating the control objective into the character's equations of motion, effectively utilizing a simulator's existing contact solver to accomplish quasi-physical control.

3.1. Solving Frictional Contacts

Most current methods for simulating frictional contacts use complementarity conditions between the contact forces and accelerations. A detailed treatment of contact handling is outside the scope of this paper, and we refer the reader to previous work [AP97]. The method described in this section will assume that we have access to a rigid body simulator that resolves contacts using a sequential impulse solver, sometimes referred to as projected Gauss-Seidel. Such solvers are used in most modern real-time simulation packages. Although we discuss contacts in terms of forces and accelerations, an equivalent formulation can be constructed with impulses and velocities.

The solver iteratively applies corrective forces to each rigid body to resolve violated contact constraints, such as sliding and intersection. The solver is initialized with the unconstrained acceleration of each body, and applies corrective forces at each contact in turn until either all contact constraints are satisfied, or the maximum number of iterations is reached. To determine the magnitude of the corrective force, the solve uses the inverse of the mass matrix of each body. In the next section, we show how our quasi-physical control objective can be integrated with such solvers.

3.2. Controlled Equations of Motions

If the contact forces are known, Equations 1-3 become a quadratic program with equality constraints, and the optimum is simply the solution of a linear system. Using this observation, we can express the optimum as an analytic linear function of the contact forces $\boldsymbol{f}_{\mathrm{c}}$. We first rewrite the objective as the sum of two quadratic forms: $\frac{1}{2}\boldsymbol{a}_k^T\boldsymbol{Q}_{\boldsymbol{a}}\boldsymbol{a}_k+\boldsymbol{a}_k^T\boldsymbol{c}_{\boldsymbol{a}}+\frac{1}{2}\boldsymbol{\tau}_k^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{\tau}_k+\boldsymbol{\tau}_k^T\boldsymbol{c}_{\boldsymbol{\tau}}$. We then note that we can express $\boldsymbol{\tau}_k$ as a linear function of $\boldsymbol{a}_k$ and $\boldsymbol{f}_{\mathrm{c}}$, resulting in a new objective $\frac{1}{2}\boldsymbol{a}_k^T\boldsymbol{Q}\boldsymbol{a}_k+\boldsymbol{a}_k^T\boldsymbol{c}+\boldsymbol{a}_k^T\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{f}_{\mathrm{c}}$, where $\boldsymbol{Q}=\boldsymbol{Q}_{\boldsymbol{a}}+\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{M}$ and $\boldsymbol{c}=-\boldsymbol{c}_{\boldsymbol{a}}-\boldsymbol{M}^T\boldsymbol{c}_{\boldsymbol{\tau}}-\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{h}$. Writing the linear goal constraint as $\boldsymbol{G}\boldsymbol{a}_k=\boldsymbol{g}$, we can express optimal accelerations and corresponding Lagrange multipliers $\boldsymbol{\eta}$ as the solution to the following linear system:

$$\underbrace{\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{G}^T\\ \boldsymbol{G} & 0\end{bmatrix}}_{\bar{\boldsymbol{M}}}\begin{bmatrix}\boldsymbol{a}_k\\ \boldsymbol{\eta}\end{bmatrix}+\underbrace{\begin{bmatrix}-\boldsymbol{c}\\ -\boldsymbol{g}\end{bmatrix}}_{\bar{\boldsymbol{h}}}=\begin{bmatrix}\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{f}_{\mathrm{c}}\\ 0\end{bmatrix}. \quad (7)$$

This linear system represents new equations of motion for the character that account for quasi-physical control. For any contact force $\boldsymbol{f}_{\mathrm{c}}$, the equations can be solved to return the corresponding optimal accelerations. To integrate our method into a rigid body simulator, we convert Equation 7 into equations of motion for each rigid link in the character's body. This is done by using the Jacobian $\boldsymbol{J}_{\mathrm{i}}$ that maps from body coordinates to the spatial coordinates of each link [Fea07]. Consider a force $\boldsymbol{f}_{\mathrm{i}}$ applied to link i. The resulting acceleration $\boldsymbol{a}_{\mathrm{k,i}}$ of link i is given by

$$\boldsymbol{a}_{\mathrm{k,i}}=-[\boldsymbol{J}_{\mathrm{i}}\ 0]\bar{\boldsymbol{M}}^{-1}\bar{\boldsymbol{h}}+[\boldsymbol{J}_{\mathrm{i}}\ 0]\bar{\boldsymbol{M}}^{-1}\begin{bmatrix}\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{J}_{\mathrm{i}}^T\boldsymbol{f}_{\mathrm{i}}\\ 0\end{bmatrix},$$

which indicates that the effective inverse mass of link i is

$$\boldsymbol{M}_{\mathrm{i}}^{-1}=[\boldsymbol{J}_{\mathrm{i}}\ 0]\bar{\boldsymbol{M}}^{-1}\begin{bmatrix}\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{J}_{\mathrm{i}}^T\\ 0\end{bmatrix},$$

and the initial unconstrained acceleration is

$$\boldsymbol{a}_{\mathrm{k,i}}^{(0)}=-[\boldsymbol{J}_{\mathrm{i}}\ 0]\bar{\boldsymbol{M}}^{-1}\bar{\boldsymbol{h}}.$$

Once per simulation step, we compute the inverse mass and initial acceleration of each body link, and allow the constraint solver to apply appropriate forces to resolve the contacts. When the solver applies a force to link i, we transform it into body coordinates by multiplying it by $\boldsymbol{M}^T\boldsymbol{Q}_{\boldsymbol{\tau}}\boldsymbol{J}_{\mathrm{i}}^T$ and add it to the current correction $\boldsymbol{f}_{\mathrm{b}}$. At the end of each solver iteration, we propagate the accumulated correction to each of the links. This is done by solving

$$\bar{\boldsymbol{M}}\begin{bmatrix}\Delta\boldsymbol{a}\\ \Delta\boldsymbol{\eta}\end{bmatrix}=\begin{bmatrix}\boldsymbol{f}_{\mathrm{b}}\\ 0\end{bmatrix}$$

to produce a correction $\Delta\boldsymbol{a}$ to the accelerations in body coordinates, and then propagating this acceleration correction to each link using their Jacobians.

4. Results

Animations generated with quasi-physical simulations are presented in the accompanying video. Our implementation runs in real time at 50 Hz on a 3.07 GHz Intel Core i7 machine. The characters used in our evaluation are shown in Figure 2. Using existing data from the CMU Motion Capture Database [CMU10], the OSU ACCAD motion capture lab [OSU10], and the University of Washington motion capture lab [UW10], we constructed three human characters with distinct skeletons. We also used a keyframed dog character from the Unity Locomotion System [Joh09], a keyframed cartoon diver, and a keyframed dinosaur character. All reference motions were situated on flat ground. Only the dinosaur character was created specifically for this project, in order to demonstrate that our approach can handle even highly non-physical motion. A variety of locomotion behaviors are demonstrated with each character, including running, jumping, cartwheeling, and crawling. The parameterized controller used in the video is described in Appendix A.

© The Eurographics Association 2012.

Figure 2: *Characters used in our evaluation. The three human characters are animated with motion capture from different sources. The dinosaur, diver, and dog characters use keyframed animations.*

The video presents results with both the quadratic program solver and the iterative two-way coupling approach, which we integrated with the open source Bullet physics engine [Bul12]. The quadratic program solver is used on all static environments, which do not otherwise require a rigid body simulator, while the iterative approach is used in the presence of simulated rigid bodies. We found that the two methods required slightly different weight parameters to achieve good results, as shown in Table 1. Since the QP solver must enforce contacts without the benefit of a physical simulation, it requires stronger end-effector terms, and a correspondingly higher weight on the pose term to preserve accurate tracking. Although the parameters differ across optimization methods, the same set of parameters are used for all motions generated with that optimization scheme.

Pushes. The accompanying video shows the responses produced by our approach to a variety of perturbations. We show how quasi-physics can produce "unconscious" responses to strong pushes while tracking a single reference clip. When combined with a kinematic controller that can change the character's velocity, quasi-physics produces plausible responses even to large perturbations.

Terrain. We also demonstrate that quasi-physics can adapt motions created on flat ground to a variety of terrains. The evaluation terrains are shown in Figure 3, and include slopes of up to 35 degrees, steps of up to 50 cm in height, and rough terrain with height variation of up to 1 meter and slopes up to 60 degrees. Since the kinematic controller is not aware of terrain variations, all responses to changing slope or steps are produced by the quasi-physical simulation. The end-effector objective E_e modifies the target positions for the feet to conform to the terrain, while the torque term E_τ produces secondary physical effects, such as swinging the arms when stepping on large steps or steep slopes.

method	$\boldsymbol{W}_\tau(i_u)$	$\boldsymbol{W}_\tau(i_a)$	$\boldsymbol{W}_p(i_q)$	$\boldsymbol{W}_e(j_c)$	ω_p, ζ_p	ω_e, ζ_e
QP	$500m^{-1}$	$m_r(i)^{-1}$	$2.5m_r(i)$	25	20, 1	20, 0.5
iterative	$100m^{-1}$	$m_r(i)^{-1}$	$1.5m_r(i)$	2.5	15, 2	20, 0.5

Table 1: *Summary of objective and PD weights for the quadratic program (QP) solver and the iterative coupled solver. Unactuated joints are denoted i_u, actuated joints are denoted i_a, all joints are denoted i_q, and end effectors are denoted j_c. $m_r(i)$ gives the mass rooted at joint i.*

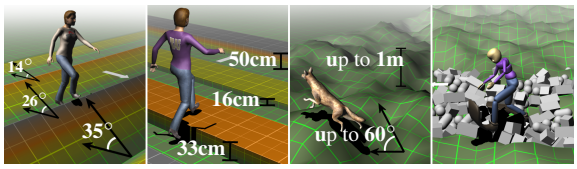

Figure 3: *Environments used in our evaluation. The environments contain steep slopes, steps, rough terrain, and simulated rubble.*

Importance of Minimization. We compare our method with a naïve PD controller to demonstrate the importance of minimizing non-physical root forces. Since the reference motion has not been made physically consistent, the PD controller must be allowed to actuate the root in order to maintain balance. However, PD control of the root does not explicitly minimize non-physical root actuation, producing unnatural responses to perturbations. Tuning the gains on such PD controllers is also difficult and time-consuming, and gains that work well for one motion may not work for another. We show how a PD controller that can track a walking motion fails when applied to sideways running, while the quasi-physical simulation reliably reproduces both motions without any parameter tuning. Although our naïve PD controller is not representative of state-of-the-art physical controllers, its purpose is to show that simply applying root forces is not sufficient to create robust controllers that appear natural and can follow any non-physical reference motion. The quasi-physical simulation appears realistic precisely because it explicitly penalizes these non-physical forces.

Physicality. We evaluate the effect of the torque term on the animation by increasing the weight on root force minimization $\boldsymbol{W}_\tau(i_u)$ by a factor of 5 (to $2500m^{-1}$), causing the simulation to emphasize physicality and heavily penalize non-physical forces. In a comparison with the usual setting of $500m^{-1}$, the emphasized physicality produces more exaggerated responses, as the character struggles to stay balanced. The weight on the root forces acts as a "knob" that can be used to tweak the magnitude of physical responses. We also compare the standard quasi-physical simulation to one in which the torque term is disabled, removing physics effects. This comparison shows how the full quasi-physical simulation adds physical detail to the animation that cannot be obtained without accounting for the equations of motion.

Two-Way Coupling. To demonstrate two-way coupling, we use the Bullet physics engine to scatter simulated rubble on rough terrain and slopes, as shown in Figure 3. Since

© The Eurographics Association 2012.

we integrate our equations of motion into the Bullet contact solver, our characters exhibit two-way interactions with this rubble. The characters successfully walk over the rubble or push it aside, while reacting appropriately to the shifting, uneven walking conditions.

5. Discussion and Future Work

We presented a method for enriching character animations with believable responses. Like physical simulation, our quasi-physical reconstruction produces physically plausible responses and interacts dynamically with other simulated objects in the environment. However, unlike methods that enforce physical consistency, our approach can animate non-physical characters and provide guarantees on the robustness of the resulting animation in the form of goal constraints. Since the proposed method does not require any preprocessing, it can act as a real-time post-process on top of any existing kinematic animation method. We demonstrate this capability by running the quasi-physical reconstruction on top of a specialized kinematic controller particularly suited for locomotion in different directions, showing that quasi-physics compliments the "conscious" responses of the controller, such as protective steps, with "unconscious" responses, such as deflection of the limbs.

Our method follows non-physical reference motions and enforces goal constraints by applying minimal non-physical forces on the root. Since these forces are penalized, they are imperceptible to the viewer, except when large perturbations are applied. Since our method can follow non-physical motions, we can add physics-based responses even to stylized, "cartoony" characters that could not previous be used with physics-based techinques. And since our goal constraints provide guarantees about the path of the character's trajectory, the reference animations do not need to be balanced or stable, and the designer need not be concerned with handling failure cases such as falling. Our method can be plugged directly into the constraint solver of an existing physics simulation package, making it easy to use with current physics simulators that are often found in interactive applications.

In future work, we hope to explore additional applications of quasi-physics. The center of mass goal constraint we use provides an intuitive interface for an application to specify the high-level behavior of the character, but our method can also readily admit any other linear goal constraint. For example, an application might choose to constrain the position of a single joint to produce "physically plausible IK," or exclude some joints from the center of mass computation to allow them to swing freely. Another exciting avenue for future work is to extend quasi-physical simulation to other objects that might interact with the character. For instance, a quasi-physical basketball player might use a combination of non-physical forces on his own root and on the ball to score a basket as dictated by the application designer, while jointly minimizing non-physical actuation on all objects.

© The Eurographics Association 2012.

6. Acknowledgments

We thank Barbara Mones for the diver character. Sergey Levine was supported by NSF Fellowship DGE-0645962.

References

[AFO05] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Pushing people around. In *Symposium on Computer Animation* (2005), pp. 56–66. 2, 3

[AP97] ANITESCU M., POTRA F. A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics 14* (1997), 231–247. 6, 7

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Eurographics Workshop on Computer Animation and Simulation* (1996), Springer-Verlag, pp. 183–197. 1, 3

[Bul12] BULLET PHYSICS:, 2012. `http://bulletphysics.org`. 8

[CMU10] CMU: Carnegie-Mellon motion capture database, 2010. `http://mocap.cs.cmu.edu`. 7

[dLMH10] DE LASA M., MORDATCH I., HERTZMANN A.: Feature-based locomotion controllers. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (2010), ACM, pp. 1–10. 2, 6

[dSAP08] DA SILVA M., ABE Y., POPOVIC J.: Interactive simulation of stylized human locomotion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (2008), ACM, pp. 1–10. 2, 6

[Fea07] FEATHERSTONE R.: *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., 2007. 4, 7

[Hor08] HORSWILL I.: Lightweight procedural animation with believable physical interactions. In *Artificial Intelligence for Interactive Digital Entertainment* (2008). 3

[Joh09] JOHANSEN R. S.: *Automated Semi-Procedural Animation for Character Locomotion*. Master's thesis, Aarhus University, 2009. 2, 3, 6, 7

[LKL10] LEE Y., KIM S., LEE J.: Data-driven biped control. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (2010), ACM, pp. 1–8. 2

[LYvdP*10] LIU L., YIN K., VAN DE PANNE M., SHAO T., XU W.: Sampling-based contact-rich motion control. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (2010), ACM, pp. 1–10. 2

[MdLH10] MORDATCH I., DE LASA M., HERTZMANN A.: Robust physics-based locomotion using low-dimensional planning. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (2010), ACM, pp. 1–8. 2

[MZS09] MACCHIETTO A., ZORDAN V. B., SHELTON C. R.: Momentum control for balance. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (2009), ACM, pp. 1–8. 2

[OSU10] OSU: Ohio State University ACCAD motion capture lab, 2010. `http://accad.osu.edu/research/mocap`. 7

[OTH02] OORE S., TERZOPOULOS D., HINTON G. E.: Local physical models for interactive character animation. *Computer Graphics Forum 21*, 3 (2002). 3

[SKL07] SOK K. W., KIM M., LEE J.: Simulating biped behaviors from human motion data. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (2007), ACM, p. 107. 2

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

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), ACM, pp. 1–7. 2

[UW10] UW: University of Washington motion capture lab, 2010. http://grail.cs.washington.edu/mocap-lab. 7

[VL95] VAN DE PANNE M., LAMOURET A.: Guided optimization for balanced locomotion. In *Eurographics Workshop on Computer Animation and Simulation* (1995), Springer, pp. 165–177. 3

[vWZR09] VAN WELBERGEN H., ZWIERS J., RUTTKAY Z.: Real-time animation using a mix of physical simulation and kinematics. *Journal of Graphics, GPU, & Game Tools 14*, 4 (2009), 1–21. 3

[WFH10] WANG J. M., FLEET D. J., HERTZMANN A.: Optimizing walking controllers for uncertain inputs and environments. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (2010), ACM, p. 112. 2

[WJM06] WROTEK P., JENKINS O. C., MCGUIRE M.: Dynamo: dynamic, data-driven character control with adjustable balance. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (2006), ACM, pp. 61–70. 2, 3, 5

[WP10] WU J.-C., POPOVIĆ Z.: Terrain-adaptive bipedal locomotion control. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (2010), ACM, pp. 1–10. 2, 6

[YL08] YE Y., LIU C. K.: Animating responsive characters with dynamic constraints in near-unactuated coordinates. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers* (2008), ACM, p. 112. 3

[YL10] YE Y., LIU C. K.: Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum 29*, 2 (2010), 555–562. 3

[YLvdP07] YIN K., LOKEN K., VAN DE PANNE M.: SIMBICON: Simple biped locomotion control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), ACM, p. 105. 2

[ZMcCF05] ZORDAN V. B., MAJKOWSKA A., CHI CHIU B. Y., FAST M.: Dynamic response for motion capture animation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 papers* (2005), ACM, pp. 697–701. 3

Appendix A: Kinematic Locomotion Control

We evaluated our method with a locomotion controller that responds to changes in the character's velocity. The controller is built from examples of the desired behavior at different speeds and in different directions. The clips are blended at runtime based on the character's velocity. When the character is pushed or receives input, the controller produces an appropriate animation for the new velocity.

Pose Retrieval The locomotion behavior is parameterized by the parameters $\boldsymbol{\theta} = (\boldsymbol{v}, \phi)$, where $\boldsymbol{v} \in \mathbb{R}^2$ is the horizontal velocity of the character, and ϕ is the fraction of the current locomotion cycle that has been completed. The three-dimensional space of poses is populated by the provided example motions, as shown in Figure 4. For each frame in each example motion, we store the pose, the change in root position and orientation from the previous frame, and the rate of change of $\boldsymbol{\theta}$, denoted $\dot{\boldsymbol{\theta}} = (\dot{\boldsymbol{v}}, \dot{\phi})$. At runtime, the motion

Figure 4: *The kinematic controller is parameterized by the character's horizontal velocity, denoted v_x and v_z, as well as the completed percentage of the locomotion cycle ϕ.*

of the character's center of mass is controlled by a particle proxy. The velocity of this particle is used to obtain $\boldsymbol{v}_k$ before each frame, which is combined with the current value of ϕ_k to obtain the current parameters $\boldsymbol{\theta}_k$. We retrieve a pose for $\boldsymbol{\theta}_k$ by finding all consecutive pairs of frames where one frame has $\phi \leq \phi_k$, and the other has $\phi \geq \phi_k$. These pairs are interpolated to obtain a set of frames with $\phi = \phi_k$, and the entire set is blended with weights given by a Gaussian kernel centered at $\boldsymbol{v}_k$. The same interpolation scheme is used to obtain the rate of change $\dot{\boldsymbol{\theta}}_k$ at $\boldsymbol{\theta}_k$.

User Control The user specifies a desired velocity $\bar{\boldsymbol{v}}$. While we could simply set $\boldsymbol{v}_k$ to $\bar{\boldsymbol{v}}$, this may cause an abrupt change in pose. Instead, we adjust $\dot{\boldsymbol{\theta}}_k$ to produce a controlled rate of change $\dot{\boldsymbol{\theta}}^\star_k$, which will be used to modify the velocity in the next frame. The controlled velocity $\dot{\boldsymbol{v}}^\star_k$ is obtained by applying a critically damped PD control law to $\boldsymbol{v}_k$ with frequency ω_v. The controlled velocity $\dot{\boldsymbol{v}}^\star_k$ is given by $\dot{\boldsymbol{v}}^\star_k = \dot{\boldsymbol{v}}_k + \Delta t \alpha_k$, and α_k is updated according to

$$\alpha_k = \alpha_{k-1} + \Delta t \left((\boldsymbol{v}_k - \bar{\boldsymbol{v}})\omega_v^2 - \alpha_{k-1}\omega_v \right).$$

To account for any mismatch between the velocity of the particle $\boldsymbol{v}_k$ and the actual velocity of the retrieved pose (for example when $\boldsymbol{v}_k$ lies outside of the convex hull of the example poses), $\dot{\phi}$ is also modified to obtain $\dot{\phi}^\star_k = \dot{\phi}_k \, ||\boldsymbol{v}_k|| \, / \, \left|\left|\boldsymbol{v}_{\boldsymbol{q}_k}\right|\right|$, where $\boldsymbol{v}_{\boldsymbol{q}_k}$ is the actual velocity of pose $\boldsymbol{q}_k$. This adjustment changes the playback speed to account for velocity mismatch. Any remaining discrepancy is handled using the quasi-physics goal constraint, which ensures that the final motion has precisely the same velocity as the particle proxy.

Proxy Particle Control After computing the controlled rate of change $\dot{\boldsymbol{\theta}}^\star_k$, we obtain the reference parameters for the next frame according to $\boldsymbol{\theta}^\star_{k+1} = \boldsymbol{\theta}_k + \Delta t \dot{\boldsymbol{\theta}}^\star_k$. A force is then applied to the particle proportional to $(\boldsymbol{v}^\star_{k+1} - \boldsymbol{v}_k)$ in order to follow the reference velocity, while still allowing deflections when the character is pushed. Advancing the particle one time step, we again obtain $\boldsymbol{v}_{k+1}$ from its velocity, and directly set ϕ_{k+1} to $\phi^\star_{k+1}$.

© The Eurographics Association 2012.