Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007)
D. Metaxas and J. Popovic (Editors)

On the Beat! Timing and Tension for Dynamic Characters

Brian Allen, Derek Chu, Ari Shapiro and Petros Faloutsos

Department of Computer Science
University of California, Los Angeles

Abstract

Dynamic simulation is a promising complement to kinematic motion synthesis, particularly in cases where simulated characters need to respond to unpredictable interactions. Moving beyond simple rag-doll effects, though, requires dynamic control. The main issue with dynamic control is that there are no standardized techniques that allow an animator to precisely specify the timing of the motion while still providing natural response to external disturbances. The few proposed techniques that address this problem are based on heuristically or manually tuning proportional-derivative (PD) control parameters and do not generalize easily.

We propose an approach to dynamic character control that is able to honor timing constraints, to provide natural-looking motion and to allow for realistic response to perturbations. Our approach uses traditional PD control to interpolate between key-frames. The key innovation is that the parameters of the PD controllers are computed for each joint analytically. By continuously updating these parameters over time, the controller is able to respond naturally to both external perturbations and changes in the state of the character.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

The acceptance of physical simulation as a standard component in interactive entertainment and "serious" games brings great possibilities to character animation. Although recent work has made significant progress in the simulation of dynamic characters, current control methods are difficult to use and require significant tuning for proper behavior. In this paper, we provide a practical solution to two troubling facets of physical characters: natural response to perturbations and precise timing control.

Small physical disturbances are a ubiquitous part of everyday life, from being bumped while queuing at the store to a shooting foul while playing basketball. Animation with natural responses to such perturbation "sells" the physicality of a character. Overly stiff or loose reactions break the viewer's suspension of disbelief.

Despite their promise, dynamic characters have seen limited use in interactive simulations outside of simple rag-doll behavior. A key impediment is that developers and animators have been asked to sacrifice a significant amount of control compared to kinematic animation. The advantage of physicality– natural response to perturbation– becomes a weakness when precisely timed motion is needed. Of course, traditional kinematic animation has no such limitation, but neither such a capacity. Control over the timing of motion is often critical for interactive applications– catching a ball, for example, is problematic if you cannot guarantee the hand's location when the ball arrives. To solve these issues, we present a method to drive a dynamic character by *physical interpolation* of key-frames.

Redefining the dynamics control problem as one of physical interpolation proves to be quite flexible. We describe three applications built on physical interpolation: pose control, key-frame animation, and tracking motion capture.

2. Overview

Our approach to dynamic character control is able to provide natural-looking motion while honoring timing constraints and providing realistic response to perturbation. We use traditional proportional-derivative (PD) feedback controllers to interpolate between key-frames in a manner analogous to classic key-frame animation. In general, PD controllers compute the magnitude of the control torque τ around a joint for

Symposium on Computer Animation 2007, San Diego, CA, August 03-04, 2007.
© 2007 ACM 978-1-59593-624-4/07/0008 $5.00

each time-step as

$$\tau = k_s(\hat{\theta} - \theta) + k_d(\hat{\omega} - \omega), \quad (1)$$

where $\hat{\theta}$ and $\hat{\omega}$ are the desired position and angular velocity, θ and ω the current position and angular velocity, and k_s and k_d the parameters that specify the trajectory of the joint's motion.

The key innovation of our approach is that the parameters of the PD equation, k_s and k_d, are updated in a principled manner for each joint axis and for each control time-step. By continuously altering the PD parameters, the controller is able to respond both to changes in the state of the character and to external perturbations, all while maintaining timing constraints.

Many prior dynamic control systems in computer graphics have used PD control, and each has needed some method for choosing the parameters. The most common approach in computer graphics has been to carefully "tune" the parameters until the desired motion trajectory is obtained.

A key draw-back to this common approach is that it simply cannot support more than one motion with timing requirements. To illustrate, suppose an animator wishes a character's arm to swing from position A to B in one second (Figure 1, upper). In typical usage, the user would need trial-and-error to find appropriate (k_s, k_d) parameters for this motion timing. If the next step of the animation required the arm to swing from B to C, the (k_s, k_d) tuned for the previous swing now also define the timing of the *second* swing as well. The user has no control over timing without breaking the previous (hard won) timing from A to B. It is important to realize that, even in this extremely simple example, there is simply no single (k_s, k_d) that will satisfy this animator's modest needs.

Indeed, the difficulty of manually choosing PD parameters does not end there. If the character wishes to perform the same arm-swing with the same timing as before, but now with a bent elbow (Figure 1, lower), the previous hand-tuned (k_s, k_d) parameters will no longer suffice. This is because the *composite inertia* of the arm has changed– the bent arm needs less torque to swing than the straightened arm.

In addition to the timing, the animator is also likely to wish to control shape of the trajectory for the arm-swing. With a PD controller, the character of the motion trajectory is determined by the ratio of $k_d : k_s$. An under-damped controller, with too low a ratio, will cause the arm to oscillate unnaturally. But increasing the ratio too much yields an over-damped controller, which moves too slowly and gives the impression of a character moving underwater. The compromise ratio produces an arm-swing that is fast, smooth and does not oscillate. This motion is said to be *critically damped*. However, the critically damped ratio between k_s and k_d depends on the moment of inertia at the joint, which in-turn depends on the limb's composite inertia. With so many varying dependencies, hand-tuning PD controllers can be a daunting task.

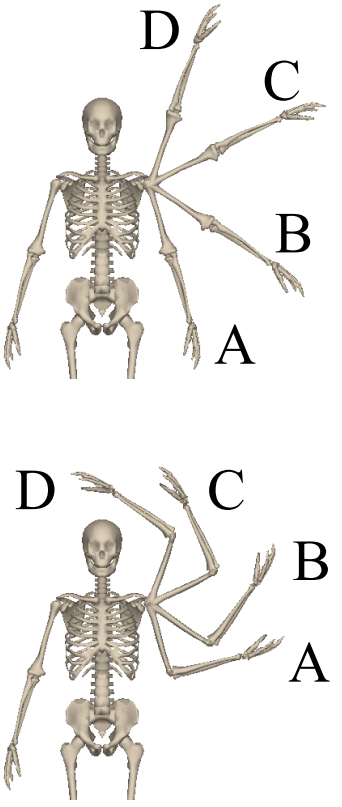

Figure 1: *The character demonstrates a simple, single-axis rotation with two arm positions. The moment of inertia at the shoulder joint changes dramatically with the bent elbow.*

Our approach is to eliminate hand-tuning and heuristic methods to determine the PD parameters, and instead compute them analytically. As was demonstrated with the simple example of a character swinging a bent arm, any method to automatically compute the PD parameters must first determine the composite inertia of all controlled links. In our approach, we descend the character's articulated hierarchy and recursively calculate and store each link's composite inertia. The implementation, described in Section 4.2, is efficient and absolutely critical to PD control with timing.

An additional complication to PD control is that the control torque is computed (1) for each joint in isolation. This introduces error, since the torque from each joint has a direct effect on every other joint in the hierarchy. To account for this effect, we use *parent-torque compensation*– that is, each joint is computed with the knowledge of the net torque of its parent joints. This approach is an efficient approximation that provides more precise control and is described in Section 4.4.

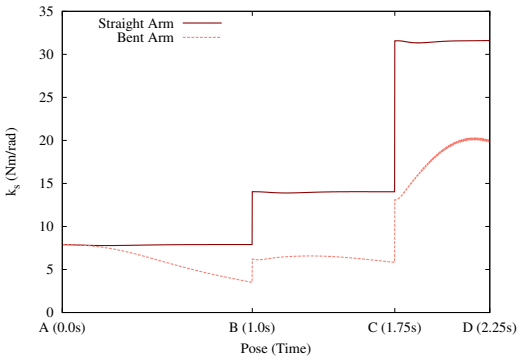

Figure 2: *The proportional parameters (k_s) used to drive the examples shown in Figure 1.*

Arbitrary trajectories, such as motion capture data, can also be tracked using this low-level controller. By sampling the motion data, a target key-frame is created and given as input to the low-level controller. Setting a key-frame based on the motion data in the near-future provides a principled means of determining muscle tension, similar to the torque-based model proposed in [YCP03].

Our approach to control provides several advantages: timing support, dynamic interpolation, and natural response with principled tension control.

A key feature of our approach is that timing control remains possible even when the character is subject to unknown perturbations. Thus our approach offers dynamic animation the reliability and specifiability familiar to kinematic animation. This combination of control and support for interaction is of critical importance to real-time games. Secondly, our method yields natural dynamic responses to perturbations. The user specifies the tension of the response by designating the time allowed before the character returns to the target trajectory. By providing tension control in human-readable units (seconds), tension control becomes intuitive and complements the general focus on precise timing.

3. Background

Currently, the dominant approaches to dynamic control for real-time animation are robotics-derived controllers [Woo98, FvdPT01a] and tracking of motion capture [ZH99]. While robotics-derived controllers have been shown to perform a surprising variety of motions [HWBO95], they also require a surprising amount of work for each new behavior. To ameliorate this burden, much effort has gone into re-using kinematic animation approaches, especially motion capture, for the control of physically simulated characters. [RGBC96] describes an approach to generate physically plausible transitions that stitch together disparate motion data. Transitions are found that minimize the applied torque, which is computed using an inverse dynamics algorithm [BP92] that determines composite inertia in a recursive manner similar to ours. [MZW99, WJM06] report that world-space error, instead of joint-space error, provides more stable tracking with a wider range of control parameters. Weinstein et al [WGF06] provides a feedback-based trajectory tracking scheme that uses a similar analytical formulation to ours to guarantee single time-step error elimination. The focus is exclusively tracking, however, and perturbation response is not considered. [ZH99, ZH02, ZMCF05] use stiff feedback controllers to track the motion capture trajectory in joint-local space. [ZMCF05, SPF03] consider circumstances where the original motion was no longer reachable due to an obstruction or large perturbation; although we detect such cases, our approach does not attempt to handle failure circumstances. [ZH02] handles reaction to perturbation by switching to heuristically determined control parameters that require human "tuning."

To provide control after a perturbation, [ZH02, ZMCF05] use an *inertia-scaled* proportional-derivative controller. This formulation scales the calculated torque by the mass of the joint's out-board link. We follow a similar approach, but compute the *composite inertia* of all links distal to the joint being controlled. Much more accurate control torque is possible if the correct composite inertia is used. Note that prior works, such as [ZH99], have suggested using composite inertia in control, but, to our knowledge, the details required to compute the composite inertia have not been provided in the computer graphics literature.

Lee and Terzopoulos [LT06] describe a biomechanical approach to tension control based on modeling multiple antagonistic muscles. Their approach produces high-quality motion with controllable levels of tension, and even demonstrates tracking of motion capture data. However, it is unclear how or even if such detailed modeling could be scaled to control entire characters in real-time.

Neff and Fiume [NF02] provide an approach to automatically computing the PD control parameters, which is the basis of our physical interpolation approach. However, their approach is limited to pose-to-pose transitions by design, and it remains unclear how it might be used to interpolate between general key-frames (with non-zero end velocities) or track motion capture.

Tension control has also been examined specifically in the context of tracking motion capture data. Zordan and Hodgins [ZH02] provide a heuristic rule for determining the proportional-derivative parameters for perturbation response. Yin et al [YCP03] suggest obtaining the tension level from an estimation of muscle activation, where activation is derived from joint torque computed using inverse dynamics. This is directly analogous to our handling of tension during motion tracking, and is a direct result of high-torques being derived from high-stiffness control parameters.

Both inverse dynamics [YCP03] and very stiff feedback controllers [ZH02] provide excellent solutions to the problem of tracking motion, but such tracking suffers from being *too* precise– natural responses to perturbations are lost. In fact, both of these approaches resort to PD control for perturbation response.

4. Physical Interpolation of Key-Frames

Our control approach is based on the ability to drive a character's joint to a specified key-frame. For our purposes, a key-frame, or "physical key-frame," is a tuple of desired joint-position $\hat{\theta}$, desired joint-velocity $\hat{\omega}$ and target time $\hat{t}$: $\gamma = (\hat{\theta}, \hat{\omega}, \hat{t})$. The control approach described in this section allows an animator to specify a set of key-frames directly analogous to traditional spline-based kinematic interpolation. Additionally, this low-level approach can be used to track existing motion trajectories, as described in Section 5.3.

The torque of the ith joint τ_i targeting the kth key-frame $\gamma_k = (\hat{\theta}_k, \hat{\omega}_k, \hat{t}_k)$ is computed using the proportional-derivative controller (1). During the controller update cycle, the PD control parameters, k_s and k_d, are calculated to match the timing of the next key-frame. The function $\psi(\cdot)$ is used to compute these parameters and is defined as

$$(k_s, k_d) = \psi(\mathbf{D}(\mathbf{q}), \gamma_k) = \psi(\mathbf{D}(\mathbf{q}), (\hat{\theta}_k, \hat{\omega}_k, \hat{t}_k)), \qquad (2)$$

where $\mathbf{D}(\mathbf{q})$ is the combined inertia tensor of the out-board body segments, and depends on the current state of the character $\mathbf{q}$. Calculating ψ requires determining the composite moment of inertia (Section 4.2), the scalar moment (Section 4.3), and compensating for the effect of parent joint's torque (Section 4.4). An example of the output of $\psi(\cdot)$ for a simple key-frame animation is provided in Figure 3.

4.1. Proportional-Derivative Parameters

The method of automatic calculation of the proportional-derivative parameters using ψ, defined in (2), is a major element of this paper's contribution. ψ provides the parameters that allow a particular joint to reach its target state in time $\hat{t}$, using the PD controller from (1). This can be written as a second-order ordinary differential equation in θ, with $\dot{\theta} = \omega$ and m as the moment of inertia about the joint axis,

$$m\ddot{\theta} - k_s(\hat{\theta} - \theta) - k_d(\hat{\omega} - \omega) = 0. \qquad (3)$$

The analytical solution provides the basis for determining ψ. In particular, the difference between the desired θ and ω and the system's actual state is reduced in the shortest time when the system is *critically damped*. This requires the parameters to satisfy $k_s^2 - 4\,m\,k_d = 0$, where m is the moment of inertia around the axis of applied torque [Bar98]. With this constraint, a simpler parameterization can be derived using the desired time $\hat{t}$ to eliminate a given fraction of the error f. A reasonable value for f is 0.9, which reduces the error to 10% in time t_d [Bar98].

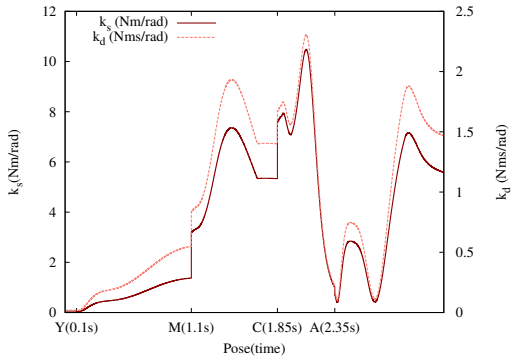

Figure 3: *The character moves through a series of poses in time to a popular song. The poses targeted are arm positions for the letters YMCA (see supplemental video). Throughout the animation, each controller update calculates $\psi(\cdot) \rightarrow (k_s, k_d)$. The calculated values for the character's shoulder joint change with each time-step to ensure correct timing throughout the animation. Manually determining these values, even for a single joint, would be difficult.*

Since the decay rate of the error is exponential when critically damped, the number n of time constants needed to eliminate the fraction f of the error is $n = -1/\ln(f)$ [BD91]. This determines the time constant, λ, needed to reduce the error adequately in time $\hat{t}$: $\lambda = \hat{t}/n$.

$$\tau = \frac{m}{\lambda^2}(\hat{\theta} - \theta) + 2\frac{m}{\lambda}(\hat{\omega} - \omega). \qquad (4)$$

This approach is directly analogous to the second-order differential equation describing the decay of constraint error in Barzel and Barr [BB88], but rather than attempting to eliminate the error in one time-step, our method reaches the target state in $\hat{t}$ seconds and provides support for a non-zero target first-derivative.

4.2. Composite Inertia Tensor

To determine the moment m in (4) the composite inertia tensor of out-board links is needed. Inertia tensors sum naturally, so $\mathbf{D}^{i..n} = \sum_{j=i}^{n} \mathbf{D}_j$ is the composite inertia tensor for joints i through n. However, this expression holds only for inertia tensors in the same coordinate system. Dynamics simulators typically store each link's inertia tensor in link-local coordinates for efficiency and convenience, so conversion to a consistent frame is required.

The recursive Newton-Euler formulation, proposed by [LWP80], is efficient [Lil93] and easy to implement. For each connecting joint in the articulated object, transform the child's inertia tensor into the parent's coordinate frame, and replace the parent's inertia tensor with the sum of parent and child. By starting the recursion at each end-effector and ensuring to accumulate over each joint once, the composite inertia for each link is compute in $O(n)$ time.

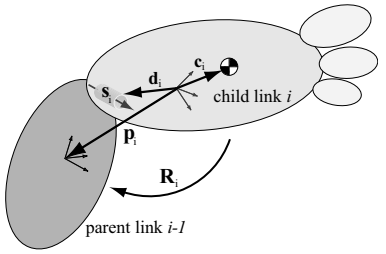

Figure 4: *This schematic shows the geometrical elements used by (5) to transform the inertia tensor of the ith child link into the parent link's $(i-1)$th coordinate frame, and by (7) to calculate the moment around joint axis $\mathbf{s}_i$. The transformation from child to parent is a translation $\mathbf{p}_i$ and rotation $\mathbf{R}_i$. In the child's link-local origin, the position vector of the center-of-mass is $\mathbf{c}_i$ and the position vector of the joint is $\mathbf{d}_i$.*

To perform the transformation from child to parent, the coordinate system transform is decomposed into a rotation matrix, $\mathbf{R}$, and the linear distance from the child's to the parent's coordinate frame $\mathbf{p}$. The position vector of the center of mass in the child's coordinate frame is $\mathbf{c}$. The child's inertia tensor $\mathbf{D}$ is transformed to its parent's coordinate frame $\mathbf{D}'$ by

$$\mathbf{D}' = \mathbf{R}\mathbf{D}\mathbf{R}^T - M\left([\mathbf{p}][\mathbf{R}\mathbf{c}] + [\mathbf{R}\mathbf{c}][\mathbf{p}] + [\mathbf{p}][\mathbf{p}]\right), \qquad (5)$$

where M is the scalar mass of the child link and $[\cdot]$ is the notation for the skew-symmetric matrix defined by $\mathbf{u}\times\mathbf{v} = [\mathbf{u}]\mathbf{v}$, or equivalently,

$$[\mathbf{u}] = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}. \qquad (6)$$

Using this expression, a composite inertia tensor ${\mathbf{D}_i}^{i..n}$ for link i can be computed recursively, from distal to proximal. The composite tensor is expressed in the ith link's coordinate frame, and combines its own and all of its out-board links' inertia, from i to the distal-end link n.

4.3. Moment of Inertia about the Joint Axis

With the composite inertia tensor of the ith joint in link-local coordinates ${\mathbf{D}_i}^{i..n}$ available, we can compute the moment of inertia about the ith joint's axis of torque $\mathbf{s}_i$ using the parallel-axis theorem as

$$m_i = \mathbf{s}_i \cdot {\mathbf{D}_i}^{i..n}\mathbf{s}_i + M_{i..n}\left\|(\mathbf{c}_i^{i..n} - \mathbf{d}_i)\times\mathbf{s}_i\right\|^2, \qquad (7)$$

where $\mathbf{c}_i^{i..n}$ is the vector from the combined center of mass and $M_{i..n}$ the combined scalar mass of the ith link and all child links, $\mathbf{d}_i$ is the vector from the link's coordinate frame to the joint, and $\mathbf{s}_i$ is a unit vector. Both $\mathbf{c}_i^{i..n}$ and $M_{n..i}$ can be computed during the same recursive traversal from end-effectors to the root that is used to compute ${\mathbf{D}_i}^{i..n}$ (see Section 4.2), keeping the time-complexity at $O(n)$.

4.4. Parent-Torque Compensation

The computation of ψ in each controller update provides a precise estimate of the torque needed to reach the target key-frame at the appropriate time. To improve its accuracy, each joint estimates how the combined torque of all inboard (parent) joints will affect it. The joint then is able to compensate for the induced acceleration on its axis of rotation, $\mathbf{s}_i$. The total angular acceleration applied to joint i in world coordinates is

$$\mathbf{a}_i = \sum_{j=0}^{i-1} \frac{\tau_j}{m_j}\mathbf{G}_j^w\mathbf{s}_j, \qquad (8)$$

where $\mathbf{G}_j^w$ is the transformation matrix from the jth link-local coordinates to world coordinates, m_j is the moment of composite inertia (7), and τ_j is the computed scalar torque around the jth joint from (4).

The required torque to compensate for the applied angular acceleration $\mathbf{a}_i$ is that acceleration projected onto the joint-axis in world coordinates,

$$\mathbf{u}_i = \mathbf{a}_i \cdot (\mathbf{G}_j^w\mathbf{s}_i). \qquad (9)$$

With the addition of this term, we obtain the final expression for the control torque of the ith joint, so (4) becomes,

$$\tau_i = \frac{m_i}{\lambda^2}(\hat{\theta}_i - \theta_i) + 2\frac{m_i}{\lambda}(\hat{\omega}_i - \omega_i) + \mathbf{a}_i \cdot (\mathbf{G}_j^w\mathbf{s}_i). \qquad (10)$$

This final torque is applied to the joint as long as $\tau_i < \tau_{max}$, i.e., as long as the computed torque is below the maximum allowed torque for the joint. If the torque exceeds τ_{max}, then the character is not strong enough to match the desired motion. A failure message can be communicated to the controlling application, and [ZMCF05, SPF03] suggest approaches to handling such conditions.

4.5. Efficiency of Control

The full process of calculating the control torque consists of two passes through articulated link hierarchy and is summarized in Algorithm 1. First, the composite inertia pass, described in Section 4.2, recursively descends from each end-effector toward the root, visiting each link once to accumulate the composite inertia for the current state, $\mathbf{D}(\mathbf{q})$. Second, the torque is computed once for each joint in (10) is computed once for each joint, accumulating the link's angular acceleration $\mathbf{a}_i$ following the hierarchy from root to end-effector.

5. Applications

5.1. Key-Frame Animation

We have described a method to physically interpolate key-frames. It is simple to integrate this low-level controller into

Algorithm 1 $ComputeControlTorque(\mathbf{q}, \gamma) \rightarrow \tau$

for all links l, descending **do**
  if $parent(l)$ exists **then**
    $\mathbf{D}_l \leftarrow \mathbf{D}_l + transformInertia(\mathbf{D}_{parent(l)}, l)$ {see (5)}
  end if
  $j = outboardJointOfLink(l)$
  $\mathbf{a}_j \leftarrow 0$
end for
for all joints i, ascending **do**
  $l \leftarrow outboardLinkOfJoint(i)$
  $(k_{si}, k_{di}) \leftarrow \psi(\mathbf{D}_l, \gamma_i)$
  $m_i \leftarrow moment(\mathbf{D}_l, i)$ {see (7)}
  $\mathbf{u}_i \leftarrow parentTorque(i, \mathbf{a}_i)$ {see (9)}
  $\tau_i \leftarrow PD(\mathbf{q}_l, m_i, k_{si}, k_{di}, \gamma_i, \mathbf{u}_i)$ {see (10)}
  $\tau_i \leftarrow \min(\tau_i, \tau_{max})$
  for all joints c in $children(l)$ **do**
    $\mathbf{a}_c \leftarrow \mathbf{a}_c + transform(\tau_i / m_i, c)$ {see (8)}
  end for
end for
return τ

a traditional key-frame animation system. As detailed in Algorithm 1, the required input to the system is the current state vector $\mathbf{q}$ and a vector of the target key-frames $(\gamma_1, \gamma_2, \ldots)$. The motion between key-frames is specified by the first derivatives $\hat{\omega}$, analogous to Hermite spline interpolation, and determined by the dynamics of PD control.

5.2. Pose Control

If a set of target key-frames specifies all the controllable degrees-of-freedom for a particular time, then that set of key-frames is said to define a *pose*. Dynamic controllers of surprising complexity [FvdPT01b, HWBO95, vKF94, YLv07] have been built by creating a finite-state machine (FSM) that controls the target pose. Our physical interpolation system supports this approach directly through a simple scripting system.

As an example of simple control, the supplementary video includes an animation of a skeleton conducting Beethoven's Fifth Symphony. The controlling script demonstrates the power of the approach– only target (pose, time) tuples are specified. It should be noted that no other parameters are provided to the system and no "hand-tuning" has taken place. Below is an excerpt of the script.

```
...
skeleton.setTargetPose( "ArmsFront", 0.3 )
skeleton.setTargetPose( "ArmsUp", 0.3 )
skeleton.setTargetPose( "Out", 0.3 )
skeleton.setTargetPose( "UpBeat", 0.4 )
skeleton.setTargetPose( "DownBeat", 0.545 )
skeleton.setTargetPose( "UpBeat", 0.545 )
...
```

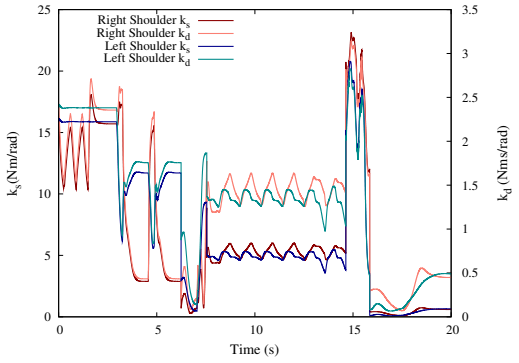

Figure 5: *The output of $\psi(\cdot)$ for the X-axis of the left and right shoulders during the conducting motion shown in the supplementary video.*

The parameters for the character's shoulder joints generated by this script are shown in Figure 5.

5.3. Tracking Motion Capture

A variety of approaches to tracking motion capture has been proposed in animation literature, with the most common being stiff PD control [ZH02, ZMCF05], and inverse dynamics [YCP03]. Our approach also uses PD control, building directly on the physical key-frame interpolation control described (Section 4).

At each control time-step, a desired key-frame γ is created by sampling the motion capture data. This key-frame contains joint velocity, which be obtained from the motion capture data by using the finite difference between frames.

5.3.1. Perturbation Response

When a perturbation to the character is detected (such as an unexpected collision), a parameter δ controls the stiffness of the response by specifying the time, in seconds, to resolve the perturbation and return the character's motion to the motion capture trajectory.

On disturbance, a new key-frame vector $\gamma_{t+\delta}$ is created by sampling the motion capture data at time $(t+\delta)$. Targeting $\gamma_{t+\delta}$ guarantees that the perturbation response will be completed by time $t+\delta$. During this δ second window, the motion capture is no longer tracked and only the new key-frame at $t+\delta$ is used.

The δ parameter allows a principled, known-units (seconds) parameter for controlling the response stiffness.

6. Results

With our method, the animator need only specify a set of key-frames containing desired position, velocity and timing information. In Figure 6 (top row), a character catches a

moving ball based on a single key-frame with timing information. In the bottom row, the character's arm responds to an external disturbance and still catches the ball at the correct time, without requiring any additional effort or information from the animator.

In Figure 7, we compare our method to a manually tuned PD controller. The two characters go through four key-frames following the popular "YMCA" dance. The manually tuned system has been adjusted to reach the "Y" key-frame at the appropriate time. As expected, the timing of the subsequent key-frames is off. It is worth noting that in the case of the manually tuned system, it took a fairly experienced student several minutes to get the timing of the "Y" key-frame correct. Using our approach, the user has only to specify the desired time of the key-frame.

In Figure 8, we compare our method to a "locally" critically damped PD controller that does not account for the composite inertia of the articulated chain. This method has been widely used, such as in [ZH02, ZMCF05]. The two characters conduct the opening movement of Beethoven's Fifth Symphony by interpolating between twelve poses. Our approach provides natural looking motion with correct timing.

6.1. Implementation

The results provided used a character model with 39 degrees-of-freedom and human-like limb-lengths, masses, and moments of inertia. We used the DANCE animation environment [SFNTH05] with the Open Dynamics Engine [Smi07] for forward-dynamics simulation and collision detection.

7. Discussion and Limitations

The described approach to dynamic character control emphasizes practical utility and natural-looking motion. Since PD control is equivalent to a spring force, using a PD controller to resolve error from perturbation is equivalent to the parallel-elastic element of the Hill-type muscle model [ZW90], providing a representation of passive neuromotor feedback [YCP03]. This approximation may not be sufficiently accurate for demanding applications. In particular, muscle stiffness is known to be a non-linear function of a variety of factors, most importantly extension length [WC00]. In addition, PD control is well-known to exhibit steady-state error in the presence of external forces (such as gravity). As our approach is based on PD control, it also suffers from such errors and key-frame targets may not be reached precisely.

Our approach for compensating for parent torque, in Section 4.4, uses only a single pass from the root to end-effectors. Although efficient, the algorithm does not take into account subsequent effects of child torques on parent joints. An $O(n^3)$ algorithm that simultaneously solves for the torque at all joints is described as Method 3 in [WO82].

An additional complication to our recursive formulation is that it assumes the character has a fixed base serving as root. This assumption suffices for upper-body motion, since the torso dominates the mass of the system and provides an approximately fixed platform. The fixed-root assumption is particularly well-suited for systems using balance-assisting root springs to stabilize the character [vL95, WJM06, SCAF07]. However, the proposed method is not appropriate for determining control parameters for support limbs.

Perturbations that are sufficiently strong and close to a target key-frame will require unnaturally large joint torques to meet the timing constraint. In these cases, we apply a per-joint limit on torque (see τ_{max} in Section 4.4). A single, time-invariant, per-joint limit does not accurately model maximum muscle exertion, since a variety of additional factors are known [WC00], such as fatigue.

8. Conclusion

We have presented an approach to controlling the timing and apparent stiffness of motion using PD control for dynamic characters. An animator specifies a set of key-frames and the associated timing information. Our system automatically computes the proportional and derivative parameters of the PD controller. The resulting parameters yield control torques that honor the timing of the provided key-frames even when the character is disturbed by external forces. The resulting character motion exhibits natural stiffness. The proposed approach is efficient, straight-forward to implement, and entirely automatic.

Acknowledgments

The authors thank the anonymous reviewers for their suggestions and comments, and Demetri Terzopoulos and Sung-Hee Lee for many helpful discussions and suggestions. BBC Radio and the Internet Archive graciously provided to the public domain the performance of Beethoven's Fifth Symphony used in the supplemental video. The work in this paper was partially supported by NSF grant No. CCF-0429983. We thank Intel Corp., Microsoft Corp., ATI Corp., and AGEIA Corp. for their generous support through equipment and software grants.

References

[Bar98] BARUH H.: *Analytical Dynamics*. McGraw-Hill, 1998.

[BB88] BARZEL R., BARR A. H.: A modeling system based on dynamic constraints. *Computer Graphics 22*, 4 (August 1988), 179–188.

[BD91] BOYCE W. E., DIPRIMA R. C.: *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*, 1st ed. Springer, 1991.

[BP92] BALAFOUTIS C., PATEL R.: *Elementary Differential Equations and Boundary Value Problems*, 5th ed. Wiley and Sons, 1992.

[FvdPT01a] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 251–260.

[FvdPT01b] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: The virtual stuntman: Dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics 25*, 6 (2001), 933–953.

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 71–78.

[Lil93] LILLY K. W.: *Efficient Dynamic Simulation of Robotic Mechanisms*, 1st ed. Kluwer Academic Publishers, 1993.

[LT06] LEE S.-H., TERZOPOULOS D.: Heads up! Biomechanical modeling and neuromuscular control of the neck. *ACM Trans. Graph. 25*, 3 (2006), 1188–1198.

[LWP80] LUH J. Y. S., WALKER M. W., PAUL R. P. C.: On-line computational scheme for mechanical manipulators. *Trans. ASME, J. Dynamic Systems, Measurement and Control 102*, 2 (1980), 69–76.

[MZW99] MATARIĆ M. J., ZORDAN V. B., WILLIAMSON M. M.: Making complex articulated agents dance. *Autonomous Agents and Multi-Agent Systems 2*, 1 (1999), 23–43.

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), ACM Press, pp. 81–88.

[RGBC96] ROSE C., GUENTER B., BODENHEIMER B., COHEN M. F.: Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 147–154.

[SCAF07] SHAPIRO A., CHU D., ALLEN B., FALOUTSOS P.: A dynamic controller toolkit. In *The 2nd Annual ACM SIGGRAPH Sandbox Symposium on Videogames* (2007).

[SFNTH05] SHAPIRO A., FALOUTSOS P., NG-THOW-HING V.: Dynamic animation and control environment. In *GI '05: Proceedings of the 2005 conference on Graphics interface* (2005), pp. 61–70.

[Smi07] SMITH R.: Open dynamics engine. http://www.ode.org/, 2007.

[SPF03] SHAPIRO A., PIGHIN F. H., FALOUTSOS P.: Hybrid control for interactive character animation. In *11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 455–461.

[vKF94] VAN DE PANNE M., KIM R., FIUME E.: Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94* (1994), pp. 208–215.

[vL95] VAN DE PANNE M., LAMOURET A.: Guided optimization for balanced locomotion. In *Eurographics Workshop on Computer Animation and Simulation '95* (1995), Terzopoulos D., Thalmann D., (Eds.), Springer-Verlag, pp. 165–177.

[WC00] WINTERS J., CRAGO P.: *Biomechanics and Neural Control of Posture and Movement*. Springer New York, 2000.

[WGF06] WEINSTEIN R., GUENDELMAN E., FEDKIW R.: Impulse-based pd control for joints and muscles. In *Sketches, Proceedings of ACM SIGGRAPH* (2006).

[WJM06] WROTEK P., JENKINS O. C., MCGUIRE M.: Dynamo: Dynamic data-driven character control with adjustable balance. In *Sandbox 06: Proceedings of the 2006 ACM SIGGRAPH Video Games Symposium* (2006).

[WO82] WALKER M. W., ORIN D. E.: Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control 104* (1982), 205–211.

[Woo98] WOOTEN W.: *Simulation of Leaping, Tumbling, Landing and Balancing Humans*. PhD thesis, Georgia Institute of Technology, 1998.

[YCP03] YIN K., CLINE M. B., PAI D. K.: Motion perturbation based on simple neuromotor control models. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), IEEE Computer Society, p. 445.

[YLv07] YIN K., LOKEN K., VAN DE PANNE M.: SIMBICON: Simple biped locomotion control. *ACM Trans. Graph.* (2007).

[ZH99] ZORDAN V. B., HODGINS J. K.: Tracking and modifying upper-body human motion data with dynamic simulation. In *In Proceedings of Computer Animation and Simulation '99* (September 1999).

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *In Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM Press, pp. 89 – 96.

[ZMCF05] ZORDAN V. B., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. *ACM Trans. Graph. 24*, 3 (2005), 697–701.

[ZW90] ZAJAC F., WINTERS J.: Modeling musculoskeletal movement systems: Joint and body segmental dynamics, musculoskeletal actuation, and neuromuscular control. *Multiple Muscle systems, JM Winters and SL-Y Woo editors. Springer Verlag, New York* (1990), 121–148.

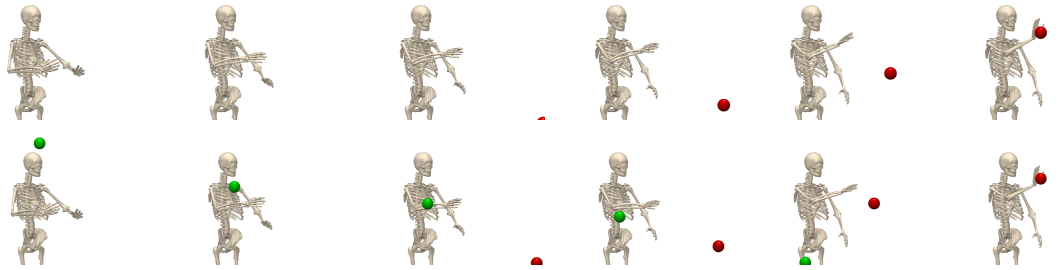

Figure 6: *A character catches a red ball at the specified time both undisturbed (top) and when disturbed by a dynamic object (bottom).*

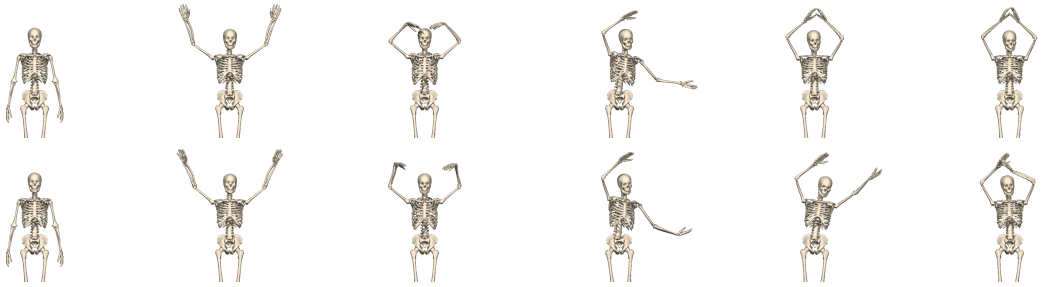

Figure 7: *Snapshots from characters performing the YMCA "dance." Our method (top) vs. PD controllers using constant, manually tuned parameters (bottom).*

Figure 8: *Snapshots from characters conducting Beethoven's 5th. Our method (top) vs. critically damped PD controllers calculated using only single-link inertia (bottom).*