Impulse Based Control of Joints and Muscles

Rachel Weinstein, Eran Guendelman, and Ron Fedkiw, *Member, IEEE*

***Abstract*— We propose a novel approach to proportional derivative (PD) control exploiting the fact that these equations can be solved analytically for a single degree of freedom. The analytic solution indicates what the PD controller would accomplish in isolation without interference from neighboring joints, gravity and external forces, outboard limbs, etc. Our approach to time integration includes an inverse dynamics formulation that automatically incorporates global feedback so that the per joint predictions are achieved. This effectively decouples stiffness from control so that we obtain the desired target regardless of the stiffness of the joint, which merely determines *when* we get there. We start with simple examples to illustrate our method, and then move on to more complex examples including PD control of line segment muscle actuators.**

***Index Terms*— proportional derivative control, torque control, muscle control, animation**

I. Introduction

OVER 20 years ago, [1], [2] used torques to control joint angles, and the notion of combining dynamics with character animation has remained exciting ever since. For example, consider the spacetime constraints animation of Luxo Jr. [3], the physics based athlete animations just before the 1996 Summer Olympics in Atlanta [4], and the composable controllers used to make a virtual stuntman [5].

With few exceptions, most systems use linear or nonlinear springs to target a desired angle. The most popular of these is proportional derivative control $\tau = -k_1(\theta - \theta_o) - k_2(\dot{\theta} - \dot{\theta}_o)$. Many authors omit $\dot{\theta}_o$ instead targeting a zero velocity, although there are notable exceptions, e.g. [4]. In spite of its popularity, PD control has many agreed upon drawbacks. For example, [6] pointed out how applying torques at one joint adversely effects others, [7], [8] criticized the large amount of tuning necessary to achieve proper gains, [9] noted the lack of system wide feedback, and [10] discussed how gravity and external forces cause errors even at equilibrium and how increasing gains to alleviate this results in overly stiff characters.

In a simple setting with no closed loops, contact or collision, etc., [6], [11] proposed using $\tau = m((\theta_o - \theta)/\Delta t - \omega)/\Delta t$ which yields θ_o exactly at the end of a forward Euler step in generalized coordinates. However, they noted that this tracks the kinematic motion too closely making the character too stiff. More generally, inverse dynamics ([12], [13]) can be used to find the torques that give any acceleration desired, although inverse dynamics does not account for the time integration scheme and methods for combating drift are required. The problem is not that we cannot track motion, but that the motions are tracked too closely. This is especially problematic when the input motion is of low quality or violates physical constraints. PD control works well because it smooths out the input motion in a desirable, physically realistic fashion.

The problem with PD control stems from locality. Animators understand how to choose gains for individual joints, but are then frustrated when the behavior of the joint in question is affected by global factors such as other joints, gravity and external forces, etc. Techniques have been proposed to alleviate these problems to some degree, e.g. [14], [15] scaled the stiffness of the gains by the moments of inertia of the outboard body, [16] recommended tuning the stiffness parameters so that the character can stand up under gravity, and [17] used inverse dynamics to calibrate gains.

We propose a new PD control technique that achieves the desired per joint behavior regardless of global effects. The key idea is to realize that the PD equations can be solved analytically to find out what the PD controller aims to achieve during the time step, and then to exactly target that state. That is, instead of targeting the input motion, we target what the individual joint controllers *would* do if they were working as designed without interference from global effects and external forces. This gives us many of the properties that make other algorithms attractive in comparison to PD, while still maintaining the PD framework. For example, our method decouples stiffness from control, and accounts for global feedback via the inverse dynamics without the need for prediction models as in [9] or the need to include extra terms as in proportional integral derivative (PID) control. With regards to the compensation provided by the integral term of PID control, our method achieves the same result but in a memoryless fashion without the need for tunable parameters.

Our method decouples stiffness from control similar to [10]. However, [10] requires estimates of the external forces which our method does not need. Also, we enjoy additional benefits such as unconditional stability. We can critically damp the errors as in model reference adaptive control [7], but our critically damped path is based on the physical second order system given by the PD equations. Moreover, we obtain the desired target regardless of the stiffness of the joint, which merely determines *when* we get there. As pointed out in section III, our method can be generalized to other controllers. Also, we could include other parameterizations including the ones given by [10]. Finally, our method can be summarized as inverse dynamics tracking of PD smoothed input motion, which is similar in spirit to using non-physical Ferguson curves to smooth between extremal target angles [8].

The PD generated torques are typically integrated via time integration, which results in drift. Thus, we take an impulse based approach to the dynamics as in [18]–[21] working with impulse and velocity as opposed to force and acceleration. This

R. Weinstein is with the Stanford University Department of Computer Science and Industrial Light + Magic. E. Guendelman is with the Stanford University Department of Bioengineering. R. Fedkiw is with the Stanford University Department of Computer Science and Industrial Light + Magic.

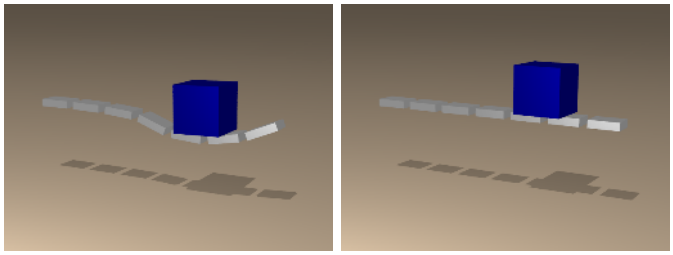

Fig. 1. **(Left) At the beginning of the simulation, the block's weight provides contact and collision forces which fight with PD resulting in the depression of the planks beneath. (Right) When the block is included in the global post-stabilization, the articulated structure automatically generates the forces required to achieve the target position. Thus, the animator can choose whether foreign objects should be heavy or effortless to lift. For example, while characters should struggle to lift a large stone, one might want them to brandish a sword with ease.**

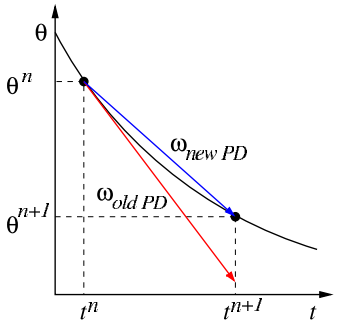

Fig. 2. **Given current data and the size of a time step, our method (blue arrow) exactly achieves the desired angle whereas traditional PD control (red arrow) overshoots. The red arrow is drawn tangent to the curve, which is the result for forward Euler time integration. Other explicit integration schemes such as Runge-Kutta overshoot differently, but the overshoots still cause inaccuracy and time step restrictions.**

also allows for more robust incorporation of external forces such as collisions and contact, giving us a hybrid between what the character is trying to do and its reaction to its environment. Figure 1 demonstrates the tradeoff between reaction to external forces and incorporation of external forces.

We use the articulated rigid body framework of [22], which maintains articulation constraints using a combination of pre-stabilization and post-stabilization. Pre-stabilization computes linear and angular impulses needed to maintain joint constraints at the end of a time step. In [22], the linear and angular impulses were determined by solving two independent systems of equations, whereas in our work we have improved upon this by solving a single, fully coupled system of equations (see section IV and appendix A). Post-stabilization uses impulses to project the joint velocities to satisfy the constraints (see also e.g. [23]). Whereas [22] treated the joints sequentially, in our work we solve for all joints simultaneously in order to incorporate global effects during inverse dynamics (see section V). Furthermore, significant generalizations of [22] would be required to address the issues of motion control and muscle actuation.

II. Other Previous Work

[24] proposed a method for slowing down a limb as it reaches the target angle, [25] merged many segments into one fixed block for efficiency (see also [26]), [27], [28] mixed high level control strategies with low level dynamics, [29]–[31] proposed interesting control strategies that work with PD control, [32] used inverse dynamics for balance and comfort, [33] showed examples of ladder climbing and push-ups (closed loops), and [15] reduced gains during contact so that characters can react.

[3] introduced spacetime constraints which impose constraints for the entire course of the animation relying on large-scale constrained optimization and sparse matrix methods. Typical objective functions minimize internal actuation, defined via an undamped spring targeting a rest angle that varies in time (as does the spring constant). These were improved by considering smaller windows with boundary conditions [34] and a hierarchical approach [35]. [36] incorporated generalized muscle forces based on PD control with the $\dot{\theta}$ term, and included a muscle smoothness term in the objective function along with a term that measures deviation from an input motion. [37] improved the approximation of muscle forces incorporating effects such as relative strengths and passive forces.

After torque actuators, the next logical step (before full-blown three-dimensional finite element analysis) are the line segment muscle actuators which control joints via lines of action as in [38], [39]. Examples include building models of the legs [40]–[42], arms [43], [44], hands [45], [46], neck [47], and body [48], [49]. There is also the musculoskeletal strand model of [50].

III. PD Control

We begin by discussing PD control in a generalized (reduced) coordinate formulation, where each coordinate represents a degree of freedom. Given a sufficiently smooth target trajectory $\theta_o(t)$ (splines, etc. can be used for smoothing if required) along with first and second derivatives of this trajectory, the PD control law specifies the generalized acceleration as

$$\ddot{\theta} = \ddot{\theta}_o - k_p(\theta - \theta_o) - k_v(\dot{\theta} - \dot{\theta}_o)$$

where k_p and k_v are the proportional (position) and derivative (velocity) gains, respectively. Note that the definition of these constants differs from that of k_1 and k_2 given in section I where torque was computed rather than acceleration. These quantities are related via $k_p = k_1/m$ and $k_v = k_2/m$ where m is a generalized moment of inertia. Many authors ignore $\dot{\theta}_o$ and $\ddot{\theta}_o$, instead targeting a zero velocity and acceleration. However, including these allows us to formulate a second order equation for the error,

$$\ddot{E} + k_v\dot{E} + k_pE = 0$$

where $E = \theta - \theta_o$. Given errors in both velocity and position, PD control drives those errors to zero while fitting naturally into a second order physics system. Moreover, similar to Baumgarte stabilization [51], the error is reduced most quickly via *critical damping* which drives E to zero with at most one overshoot. This is achieved by setting $k_v = 2\sqrt{k_p}$ reducing

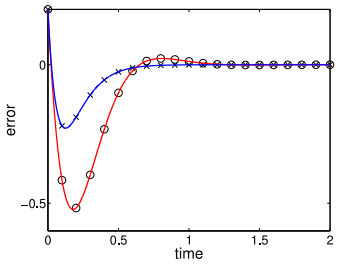

Fig. 3. **The red curve shows the results obtained using our method for k_p and k_v along with forward Euler time integration. The method converges to the analytic solution of the PD equations with $k'_p = k_p/2$ and $k'_v = k_v/2$ depicted in black circles. The blue curve shows our method with $\hat{k}_p = 2k_p$ and $\hat{k}_v = 2k_v$, which converges to the solution of the PD equations with k_p and k_v as desired (black x's).**

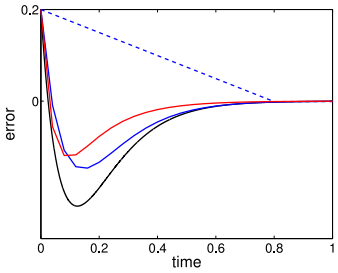

Fig. 4. **Comparison of our method (blue), traditional PD control (red) and the analytic solution (black) for a medium size time step where errors can be seen. Note that for smaller time steps, all three curves lie on top of each other. We also show the results of our method for an extremely large time step (dotted blue) where the standard force based PD control method becomes unstable and diverges. Our method removes almost all the error in one (rather big) time step as it targets the analytic solution.**

the choice of gains to a one-dimensional family parameterized by k_p. Increasing k_p causes the trajectory to be tracked more strongly. A key point here is that the trajectory is followed using a second order system compatible with the physics, in contrast to Ferguson curves or other interpolation schemes.

We exploit the fact that we can integrate analytically to obtain closed form expressions for both the error and its derivatives. In the critically damped case, the roots of the quadratic equation $r^2 + k_v r + k_p = 0$ are $r_1 = r_2 = -\sqrt{k_p}$, and we obtain

$$\begin{aligned} E(t) &= (c_1 + c_2 t)e^{-\sqrt{k_p}t} \\ \dot{E}(t) &= \left(-\sqrt{k_p}(c_1 + c_2 t) + c_2\right) e^{-\sqrt{k_p}t} \end{aligned}$$

where $c_1 = E(0)$ and $c_2 = \dot{E}(0) + \sqrt{k_p}E(0)$. Thus, the exact solution at the end of the time step is obtained by evaluating E and $\dot{E}$ at Δt, i.e. $E(\Delta t)$ and $\dot{E}(\Delta t)$. The analytic solution to the PD equations at end of the time step is therefore $\theta_E = \theta_o(\Delta t) + E(\Delta t)$ and $\dot{\theta}_E = \dot{\theta}_o(\Delta t) + \dot{E}(\Delta t)$. Note that in our impulse based approach the target acceleration $\ddot{\theta}_o$ is never used or required, even though it is accounted for in the equations. We use simple Euler time integration, which for a single degree of freedom is $\theta^{n+1} = \theta^n + \Delta t \omega^{n+1}$. Thus, if the analytically determined exact solution at the end of the time step is given by θ_E, then setting our angular velocity based on the secant to the curve (blue arrow in Figure 2) gives

$$\hat{\omega} = \frac{\theta_E - \theta^n}{\Delta t}$$

which guarantees that we achieve the exact position at the end of the Euler integration step.

Since θ_E is the exact solution at t^{n+1}, we can Taylor expand θ_E about t^n and substitute it into our equation for $\hat{\omega}$ to obtain

$$\begin{aligned} \hat{\omega} &= \frac{(\theta^n + \Delta t \omega^n + \frac{(\Delta t)^2}{2}\alpha^n + O(\Delta t^3)) - \theta^n}{\Delta t} \\ &= \omega^n + \frac{\Delta t}{2}\alpha^n + O(\Delta t^2) \end{aligned}$$

where α is angular acceleration. The $\frac{\Delta t}{2}$ illustrates that our angular velocity $\hat{\omega}$ lives most naturally at $t^{n+\frac{1}{2}}$, i.e. it is the velocity at the half time step, $\omega^{n+\frac{1}{2}}$, up to $O(\Delta t^2)$. Thus, it is straightforward to use central time differencing (e.g. see [52]),

$$\begin{aligned} \omega^{n+\frac{1}{2}} &= \omega^n + \frac{\Delta t}{2}\alpha^n \\ \theta^{n+1} &= \theta^n + \Delta t \omega^{n+\frac{1}{2}} \\ \omega^{n+1} &= \omega^{n+\frac{1}{2}} + \frac{\Delta t}{2}\alpha^{n+1} \end{aligned}$$

by replacing $\omega^{n+\frac{1}{2}}$ with $\hat{\omega}$ obviating the need for the first of these three equations (i.e., the one that determines $\omega^{n+\frac{1}{2}}$). However, we use the method of [20] (that robustly treats contact and collision, e.g. removing contact chatter) which requires a time discretization of the form

$$\begin{aligned} \omega^{n+1} &= \omega^n + \Delta t \alpha^n \\ \theta^{n+1} &= \theta^n + \Delta t \omega^{n+1} \end{aligned}$$

where our modification is to replace ω^{n+1} with $\hat{\omega}$. Since $\hat{\omega}$ naturally lives at $t^{n+\frac{1}{2}}$, the resulting position update, $\theta^{n+1} = \theta^n + \Delta t \hat{\omega}$, resembles the second of the three central differencing equations. Whereas central differencing has a third equation to integrate $\omega^{n+\frac{1}{2}}$ to ω^{n+1}, it is not clear how to incorporate this into the method of [20] (which we use), and we therefore omit it. As a result, in the limit as $\Delta t \to 0$, only half of the acceleration $-k_p E - k_v \dot{E}$ is accounted for, which is equivalent to using PD control with gains $k'_p = k_p/2$ and $k'_v = k_v/2$ implying that $k'_v \neq 2\sqrt{k'_p}$ (the system is underdamped, not critically damped).

This problem is trivially corrected by applying our method to a set of PD equations with twice the acceleration. That is, we apply our secant time integration method to the modified system,

$$\ddot{E} + \hat{k}_v \dot{E} + \hat{k}_p E = 0$$

with $\hat{k}_v = 2k_v$ and $\hat{k}_p = 2k_p$. This is an *overdamped* system with roots $r = (-\hat{k}_v \pm \sqrt{\hat{k}_v^2 - 4\hat{k}_p})/2$, and an analytic solution of

$$\begin{aligned} E(t) &= c_1 e^{r_1 t} + c_2 e^{r_2 t} \\ \dot{E}(t) &= r_1 c_1 e^{r_1 t} + r_2 c_2 e^{r_2 t} \end{aligned}$$

where $c_1 = (r_2 E(0) - \dot{E}(0))/(r_2 - r_1)$ and $c_2 = (\dot{E}(0) - r_1 E(0))/(r_2 - r_1)$. The solution of this overdamped system at the end of the time step, $E(\Delta t)$,

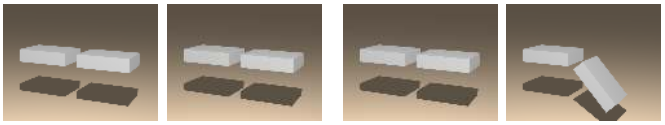

Fig. 5. **In the first two images, our method (first image) and traditional PD control (second image) perform equally well in zero gravity. However, the next two images show the results with gravity where our method (third image) still exactly reaches the desired position while traditional PD control (fourth image) fails to achieve it.**

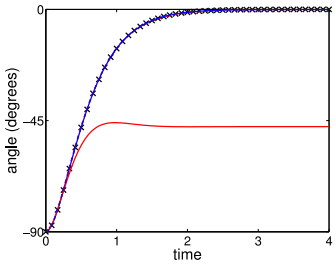

Fig. 6. **Plots of the angle for the examples shown in Figure 5 as compared to the analytic solution (black x's). Our method matches the analytic solution without gravity (blue dashed) and with gravity (blue), whereas traditional PD control matches it without gravity (red dashed) but fails to with gravity (red).**

is used to find $\theta_E = \theta_o(\Delta t) + E(\Delta t)$ which is used as the exact solution in $\hat{\omega} = (\theta_E - \theta^n)/\Delta t$. Incorporating this $\hat{\omega}$ into our time integration scheme yields a solution with half the acceleration, i.e. it yields the solution to the PD equations with gains of $k'_p = \hat{k}_p/2$ and $k'_v = \hat{k}_v/2$ as desired. Figure 3 illustrates this approach. Note that the correct critically damped formulation reaches zero quicker than the underdamped case. Moreover, Figure 4 shows that our scheme remains unconditionally stable driving the error to zero as desired even for large time steps.

Finally, note that the results of this section are not restricted to PD control. Our secant based approach can be applied to one's favorite control method as long as that method satisfies a few basic principles, such as being able to determine (or even sketch by hand) the desired behavior for a single degree of freedom. The desired control curve would be targeted using inverse dynamics and extended to situations such as multiple joints just as we do for PD control.

IV. Dynamics

Following [22] we process collisions, post-stabilize velocities by projecting them onto a constraint satisfying manifold, integrate velocity forward in time (add gravity, etc.), and post-stabilize again. Then we apply their combined contact processing and pre-stabilization method that evolves the rigid body positions and orientations in a manner that satisfies both contact conditions and articulation constraints. Notably, we improve pre-stabilization by solving a *single* nonlinear equation $\mathbf{f}(\mathbf{j}, \mathbf{j}_\tau) = \mathbf{0}$ (with a 7×6 Jacobian) for the linear and angular impulses, $\mathbf{j}$ and $\mathbf{j}_\tau$, that satisfy both positional and angular articulation constraints simultaneously, rather than using two separate equations (see appendix A). This improves pre-stabilization, since it allows for quicker convergence by accounting for the cross-coupling between linear and angular impulses. Then post-stabilization is applied to the velocities once again.

Since PD control is typically integrated into velocity as a force, we incorporate our method into the post-stabilization step that immediately follows the velocity update. Post-stabilization solves for the impulses $\mathbf{j}$ and $\mathbf{j}_\tau$ that maintain the constraints on a joint by joint basis. We define the 6×6 matrix

$$K_b = \begin{bmatrix} \delta & \mathbf{r}_b^{*T} \\ 0 & \delta \end{bmatrix} \begin{bmatrix} m_b^{-1}\delta & 0 \\ 0 & I_b^{-1} \end{bmatrix} \begin{bmatrix} \delta & 0 \\ \mathbf{r}_b^{*} & \delta \end{bmatrix}$$

where b refers to an arbitrary body, δ is the identity matrix, I_b is the inertia tensor of body b, and $\mathbf{r}_b$ is a vector from b's center of mass to the joint location (where impulses are applied). This allows us to compactly write the post-stabilization equations given in [20], [22] as

$$(K_p + K_c) \begin{pmatrix} \mathbf{j} \\ \mathbf{j}_\tau \end{pmatrix} = \begin{pmatrix} \Delta \mathbf{u}_{rel} \\ \Delta \omega_{rel} \end{pmatrix} \tag{1}$$

where p and c refer to the joint's parent and child, and $\Delta \mathbf{u}_{rel}$ and $\Delta \omega_{rel}$ are the changes in relative (parent minus child) linear and angular velocities at the joint caused by the given impulses. Given desired velocity changes, the impulses are found by inverting the symmetric positive definite $K_p + K_c$.

The desired values for $\Delta \mathbf{u}_{rel}$ and $\Delta \omega_{rel}$ are determined by a combination of joint constraints and PD control. For example, for a hinge joint with one rotational degree of freedom, in the PD controlled direction we want the secant-based $\hat{\omega}$ computed as described above, while a zero velocity is targeted in the other two angular dimensions as well as for the fully constrained positional degrees of freedom (projecting onto the constraint manifold as in standard post-stabilization).

PD actuation of a 3 degree of freedom point joint is only slightly more involved. In this case, our target trajectory θ_o is an orientation and $\dot{\theta}_o$ is an angular velocity. A joint consists of joint frames attached to both the parent and child. If we let J_p and J_c represent the orientation of the parent and child attached joint frame, then we define the joint's orientation to be $J_c^{-1} J_p$ (parent joint orientation with respect to the child). Define $\mathbf{v}^n$ to be the rotation vector corresponding to the rotation $J_c \theta_o^n J_p^{-1}$, i.e. a rotation of $|\mathbf{v}^n|$ radians about the unit vector $\hat{\mathbf{v}}^n$. This is a world space representation of the rotation error between the current joint orientation and the current target orientation (analogous to $\theta^n - \theta_o^n$). Setting $E(0) = |\mathbf{v}^n|$ and $\dot{E}(0) = (\dot{\theta}^n - \dot{\theta}_o^n) \cdot \hat{\mathbf{v}}^n$, we solve for the scalar quantity $E(\Delta t)$ as usual and set the desired joint angular velocity to

$$\hat{\omega} = \frac{E(\Delta t) - |\mathbf{v}^{n+1}|}{\Delta t} \hat{\mathbf{v}}^{n+1}$$

where $\mathbf{v}^{n+1} = J_c \theta_o^{n+1} J_p^{-1}$ represents the error between the joint's *current* orientation and the *updated* trajectory orientation (i.e. where it wants to be). This formula matches our usual one since $|\mathbf{v}^{n+1}|$ represents $\theta^n - \theta_o^{n+1}$ so the numerator looks like $\theta_E - \theta^n$. In general, the current angular velocity ω^n may not be parallel to $\mathbf{v}^{n+1}$ (or even to $\mathbf{v}^n$). This means that the above simplification to one dimension does not adequately address the other two dimensions (perpendicular to $\hat{\mathbf{v}}^{n+1}$). The full multi-dimensional application of PD control should

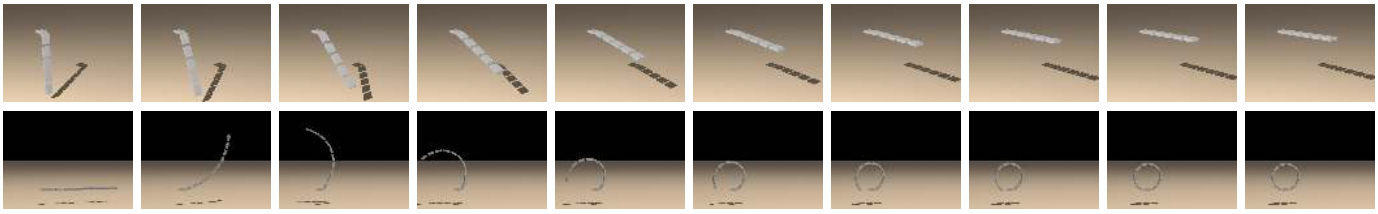

Fig. 7. **(Top) Even with multiple joints, our method can attain and hold the desired horizontal state without difficulty. (Bottom) A similar example with a different target state. We stress that all joints are controlled locally and there is no need for scaling joints based on outboard inertia tensors, or for estimates of external forces and gravity.**

be applied to all three dimensions, although our specific one-dimensional restriction lumps all positional errors into that one dimension leaving positional errors of zero in the other two dimensions. While there are no positional errors in the remaining two dimensions, velocity errors do exist, similar to a one-dimensional case with an overshoot where the positional errors are temporarily zero. We can apply our technique to these other two dimensions by adding to $\hat{\omega}$ an extra term $e^{-\hat{k}_v \Delta t}\omega^{n,\perp}$ where $\omega^{n,\perp}$ is the component of ω^n orthogonal to $\hat{\mathbf{v}}^{n+1}$.

If a joint has degrees of freedom that are not constrained or actuated, we solve a projected version of equation 1 so that the dynamics in those degrees of freedom are not adversely affected, i.e.

$$P^T(K_p + K_c)P\hat{\mathbf{j}}^P = P^T \begin{pmatrix} \Delta\mathbf{u}_{rel} \\ \Delta\omega_{rel} \end{pmatrix}$$

where P is a matrix whose columns form a basis for the space of allowable impulses, and $\hat{\mathbf{j}}^P = P^T\hat{\mathbf{j}}$ is the projected linear and angular impulse ($\hat{\mathbf{j}}$ representing the concatenation of $\mathbf{j}$ and $\mathbf{j}_\tau$). The joint velocities (on the right hand side) are also projected so that the velocity in the free directions does not influence our system. The full six-dimensional spatial impulse is then recovered using $P\hat{\mathbf{j}}^P$.

We demonstrate single joint post-stabilization with a few simple examples as shown in Figure 5. Given zero gravity, or no external forces, both our algorithm and traditional PD control reach the desired target angle of zero degrees. However, when gravity is introduced, traditional PD control fails to bring the joint to its desired state while ours accomplishes this. As the gains are increased, traditional PD performs better (although with a more severe time step restriction), but this is unnecessary for our algorithm which performs well with any gains allowing gains to be set based on desired stiffness rather than for motion tracking purposes. Figure 6 plots the angles of the joints with respect to time.

V. Inverse Dynamics

Extending PD control to multiple joints significantly complicates the situation, because each joint generates forces that interfere with neighboring joints. We use inverse dynamics to alleviate this problem solving for the impulses that take surrounding joints into account. This is accomplished by extending post-stabilization from a joint by joint approach to a global approach for the entire articulated rigid body. While the previous section gives the post-stabilization equation for a single joint, the global framework requires a matrix A relating the impulses at all joints to the change in velocities at all joints. A is an $n \times n$ block matrix (n the number of joints) composed of 6×6 blocks. In order to describe the contents of a block in A, we generalize the K_b matrix from section IV to $K_b(i,k)$:

$$K_b(i,k) = \chi_i(b)\chi_k(b) \begin{bmatrix} \delta & \mathbf{r}_{b,i}^{*T} \\ 0 & \delta \end{bmatrix} \begin{bmatrix} m_b^{-1}\delta & 0 \\ 0 & I_b^{-1} \end{bmatrix} \begin{bmatrix} \delta & 0 \\ \mathbf{r}_{b,k}^{*} & \delta \end{bmatrix}$$

The "connectivity" factor $\chi_\alpha(b)$ is defined to be $+1$, -1, or 0 depending on whether b is a parent of, child of, or not connected to joint α, respectively. In $K_b(i,k)$, $\chi_k(b)$ is used to indicate whether the *action* or *reaction* (negated) impulses are being applied to b at k, and $\chi_i(b)$ corresponds to the fact that we are measuring parent velocity *minus* child velocity at joint i. Block (i,k) of A can then be written succinctly as

$$A(i,k) = \sum_b K_b(i,k)$$

One can think of $A(i,k)$ as relating the impulses $\mathbf{j}^k$ and $\mathbf{j}_\tau^k$ applied at joint k to the change in relative velocities $\Delta\mathbf{u}_{rel}^i$ and $\Delta\omega_{rel}^i$ at joint i. The global system can be written

$$A\hat{\mathbf{j}} = \Delta\hat{\mathbf{u}}$$

where impulses and velocities have been concatenated into vectors $\hat{\mathbf{j}}$ and $\Delta\hat{\mathbf{u}}$. The values of $\Delta\hat{\mathbf{u}}$ come from a combination of joint constraints and PD actuation as in the single joint case (see section IV). Furthermore, projection matrices may be used to reduce the system if any unconstrained and unactuated degrees of freedom remain.

Note that A is both symmetric ($K_b(i,k) = K_b(k,i)^T$) and sparse ($A(i,k)$ is only non-zero when joints i and k share a common body). Symmetry is guaranteed regardless of how parent and child labels are assigned at a joint. Unlike the case for a single joint where $A = K_p + K_c$ is positive definite, A may in general be singular. Thus, we first factor A using QR factorization. Using column pivoting we can ensure R has the form

$$R = \begin{bmatrix} B & S \\ 0 & \epsilon \end{bmatrix}$$

where B is an upper triangular square matrix with well-conditioned, non-zero diagonal entries, and ϵ is zero up to some user chosen tolerance. If P is the pivot matrix ($AP = QR$), then our new system is $RP^T\hat{\mathbf{j}} = Q^T\Delta\hat{\mathbf{u}}$. To simplify notation, let $\mathbf{x} = P^T\hat{\mathbf{j}}$ and $\mathbf{b} = Q^T\Delta\hat{\mathbf{u}}$. Then these vectors can be decomposed to match the blocks of R, yielding

$$\begin{bmatrix} B & S \\ 0 & \epsilon \end{bmatrix} \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_S \end{pmatrix} = \begin{pmatrix} \mathbf{b}_B \\ \mathbf{b}_\epsilon \end{pmatrix}$$

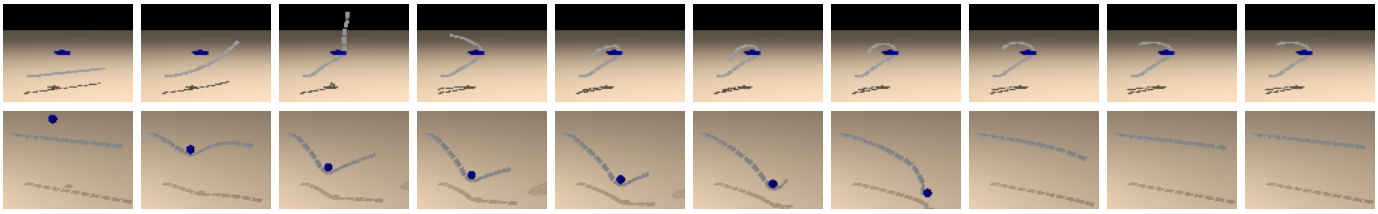

Fig. 8. **(Top) Our PD control method allows for interesting object interaction. Here, the articulated object struggles to reach its target when obstructed by an immovable foreign object. (Bottom) A collision with a ball disturbs the target position, after which it recovers gracefully. Our separation of stiffness from control allows the object to be pushed very far from its target by the ball, while still recovering without adverse effects from gravity.**

Since B is non-singular by construction, the first equation, $B\mathbf{x}_B + S\mathbf{x}_S = \mathbf{b}_B$ has a family of solutions which can be parameterized by $\mathbf{x}_S$, i.e.

$$\mathbf{x} = Z\mathbf{x}_S + \hat{\mathbf{b}}$$

where $Z = \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix}$, and $\hat{\mathbf{b}} = \begin{pmatrix} B^{-1}\mathbf{b}_B \\ \mathbf{0} \end{pmatrix}$. In order to single out a particular solution, we solve a least squares problem in $\mathbf{x}_S$ which aims to minimize both the norm of the solution $\mathbf{x}$ as well as the residual of the remaining equations $\epsilon\mathbf{x}_S = \mathbf{b}_\epsilon$. That is, we solve

$$\min_{\mathbf{x}_S} \left(\|Z\mathbf{x}_S + \hat{\mathbf{b}}\|_2^2 + \|\epsilon\mathbf{x}_S - \mathbf{b}_\epsilon\|_2^2 \right)$$

with corresponding normal equations of

$$(Z^T Z + \epsilon^T \epsilon)\mathbf{x}_S = -Z^T\hat{\mathbf{b}} + \epsilon^T\mathbf{b}_\epsilon.$$

The first term in the objective function minimizes the norm of $\mathbf{x}$, which ensures that we satisfy the first equation, $B\mathbf{x}_B + S\mathbf{x}_S = \mathbf{b}_B$, with minimal impulses. In order to account for the effect of the second term, we consider two cases. In the case where ϵ is zero, the original system is singular, and the objective function is simply offset by the constant $\|\mathbf{b}_\epsilon\|_2^2$ which is nonzero only if the original target $\Delta\hat{\mathbf{u}}$ is unachievable. Thus, in this case, the least squares problem still yields a solution with minimal impulses. In the case where ϵ is nonzero, the combination of the two terms in the objective function means that we try to minimize impulses while also minimizing errors in the second equation, $\epsilon\mathbf{x}_S = \mathbf{b}_\epsilon$. Weights can be added to the two terms to control the relative importance of these two objectives, and also to account for different units of measurement used for the two terms. We also note that the above is readily adjusted to minimize $\mathbf{x}$ using a weighted norm $\|W^{1/2}\mathbf{x}\|_2^2$ by replacing Z with $W^{1/2}Z$ and $\hat{\mathbf{b}}$ with $W^{1/2}\hat{\mathbf{b}}$ in the normal equations.

Figure 7 shows two articulated chains with gravity. Since global post-stabilization gives global feedback, joints do not fight each other and instead move smoothly towards the target state. Moreover, we only specified a single final state with no in-betweens, and still achieved characteristically smooth and natural PD style motion with our method. Furthermore, we can adjust k_p without concern for gravity or outboard inertia tensors. Besides obtaining smooth motion, an advantage of dynamic controllers (versus kinematic) is response to unanticipated forces. In the top row of Figure 8, the planks attempt to curl (as in the bottom of Figure 7) while obstructed by an immovable object. Figure 1 (left) shows an articulated chain of planks targeting a horizontal state while supporting the weight of a disjoint block. This mimics a character easily moving their own limbs around, but struggling to lift a foreign object. In the right figure, we incorporate the block into the global post-stabilization to illustrate that our PD control can fully compensate for its weight if desired.

We still iterate for pre-stabilization (as in [22]) but for post-stabilization we have a global linear system, which is sparse and solved for using a direct method. This may be slower than [22] for large systems but typically characters are not very big, and each character could be solved for independently and then coupled through external contact and collision. We also support iterative post-stabilization as in [22] which can be used in place of a global solve for large systems such as the net as seen in Figure 10.

VI. Inverse Muscle Actuation

Line segment muscles differ from PD actuation in a number of ways. First, while PD actuation impulses may be chosen from a whole subspace of possible directions, muscle actuation is strictly limited to lie along the muscle's line of action. Additionally, multiple muscles may cross a given joint (redundancy) and multiple joints may be crossed by the same muscle (e.g., the sartorius crosses both the hip and knee joints). Finally, bounds on non-dimensionalized muscle activation (between 0 and 1) induce bounds on the range of muscle force. While our method solves for muscle actuation (impulse) directly as proof of concept, musculotendon force is linear in activation in a Hill-type model making the extension to this case straightforward.

We augment our post-stabilization method to include muscle actuation by writing it in the following form

$$\begin{bmatrix} \hat{A}_1 & \hat{A}_2 \end{bmatrix} \begin{pmatrix} \hat{\mathbf{j}} \\ \hat{\mathbf{j}}_m \end{pmatrix} = \Delta\hat{\mathbf{u}} \tag{2}$$

where $\hat{\mathbf{j}}$ are the usual joint constraint and actuation impulses, while $\hat{\mathbf{j}}_m$ are the muscle impulses with $j_{m,k}$ the impulse applied by muscle k. $\hat{A}_1$ is constructed analogously to matrix A from post-stabilization, but will be a tall rather than square matrix in general. $\hat{A}_2$ captures the effect of muscle actuation on joint velocities. In similar style to $K_b(i,k)$, we can write the factor relating a muscle actuation impulse $j_{m,k}$ directed positively along the line of action $\hat{\mathbf{l}}_{m,k}$ and applied at $\mathbf{r}_b$ on the velocity at joint i as

$$\chi_i(b) \begin{bmatrix} \delta & \mathbf{r}_{b,i}^{*T} \\ 0 & \delta \end{bmatrix} \begin{bmatrix} m_b^{-1}\delta & 0 \\ 0 & I_b^{-1} \end{bmatrix} \begin{pmatrix} \hat{\mathbf{l}}_{m,k} \\ \mathbf{r}_b \times \hat{\mathbf{l}}_{m,k} \end{pmatrix}$$

This is a 6×1 vector which we project to the constrained and controlled velocity components of joint i. For a given muscle k, column k of $\hat{A}_2$ is formed by concatenating these vectors for all of the constrained and controlled joint velocities and summing across all attachments for that muscle. Note that a single muscle passing through a number of via points has multiple lines of action, but only the linear segments connecting via points on *different* bones result in a net force.

As discussed in the last section, QR with pivoting can be applied to $\hat{A}_1$ to subsequently reduce the size of the system. However, for the sake of exposition, we assume that $\hat{A}_1$ is full column rank and that its top square block $\hat{A}_{1,1}$ is invertible so that we can reduce the system by multiplying equation 2 with

$$Z = \begin{bmatrix} \hat{A}_{2,1}\hat{A}_{1,1}^{-1} & -I \end{bmatrix}$$

where $\hat{A}_{2,1}$ is the bottom block of $\hat{A}_1$. Since $Z\hat{A}_1 = 0$ this leaves us with an equation of the form

$$\tilde{A}\hat{\mathbf{j}}_m = \tilde{\mathbf{b}} \tag{3}$$

where $\tilde{A} = Z\hat{A}_2$ and $\tilde{\mathbf{b}} = Z\Delta\hat{\mathbf{u}}$. A common approach is to minimize $\|\hat{\mathbf{j}}_m\|_2^2 + \|\tilde{A}\hat{\mathbf{j}}_m - \tilde{\mathbf{b}}\|_2^2$ where the first term is added to minimize muscle activation, especially since redundant muscles can cause $\tilde{A}$ to be rank deficient. However, this approach will not attain the solution to equation 3 even when it exists, e.g. if equation 3 was the scalar equation $j = 1$, adding $j = 0$ to minimize j gives a least squares solution of $j = 1/2$. This means that the trajectories we worked so hard to formulate won't be achieved, even when the impulses necessary to follow these trajectories are feasible. Instead, our approach solves equation 3 exactly to the degree possible, and minimizes any unattainable equations (over-determined subsets) along with the redundant muscle activations. Thus, following section V, we use QR with pivoting to convert equation 3 to the form

$$\begin{bmatrix} B & S \\ 0 & \epsilon \end{bmatrix} \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_S \end{pmatrix} = \begin{pmatrix} \mathbf{b}_B \\ \mathbf{b}_\epsilon \end{pmatrix}$$

yielding the final form of the optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{x}\|_2^2 + \|\epsilon\mathbf{x}_S - \mathbf{b}_\epsilon\|_2^2 \\ \text{subject to} \quad & B\mathbf{x}_B + S\mathbf{x}_S = \mathbf{b}_B \text{ and } j_{m,k}^{min} \le j_{m,k} \le j_{m,k}^{max} \end{aligned}$$

noting that $\mathbf{x}_B$ and $\mathbf{x}_S$ are simply a permutation of the components of $\hat{\mathbf{j}}_m$.

To summarize our approach, given a target motion (specifically, target joint velocities), it may or may not be achievable with the given set of actuators, so we use QR factorization to instead target the achievable motion closest to it in a least squares sense. Then we solve a constrained optimization problem which achieves this target motion while minimizing muscle impulses and ensuring they stay within prescribed limits. Visually, the only compromise is in the first part – projecting to an achievable motion. The rest is just a choice among different muscle activations, all of which achieve the same motion, and which can be augmented to prefer some muscles over others using a weighted norm.

We solve this quadratic programming problem using a two-phase, active set approach. First, a feasible point (satisfying both equality and inequality constraints) is found by solving a linear programming problem with an objective designed to drive the iterates from vertex to vertex on the manifold of active constraints and towards the feasible region. Once a feasible point is found, we minimize the quadratic objective with quadratic programming. This iterative method moves towards a minimum by taking steps consistent with the constraints while always staying within the feasible region.

Fig. 9. **A skeleton swinging a mace demonstrates the ability to mix controlled objects with purely dynamic objects.**

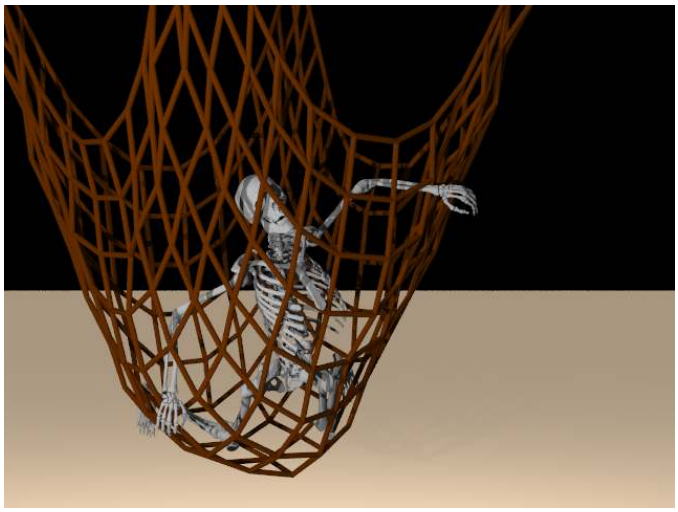

Fig. 10. **A skeleton caught in a net illustrates the ability to mix our global post-stabilization (for the skeleton) with the iterative method (for the net).**

VII. Examples

We created a skeleton from the Visible Human data set. Joints for the upper body were created from [53], while the joints for the lower body and individual digits were generated by hand. The skeleton contains 74 joints with a total of 118 degrees of freedom including fully articulate fingers, toes and vertebrae. We generated 228 muscles for the skeleton from SIMM data derived from [54], [55]. In order to demonstrate the ability to mix controlled and purely dynamic joints, we animated a skeleton swinging a mace. The arm movement is defined by an analytic function targeted via our PD control, while the mace is a series of purely dynamic, freely rotating point joints. The simulation time was about a minute per frame. We demonstrate the ability to handle closed loops with our net example (Figure 10), which contains 960 joints

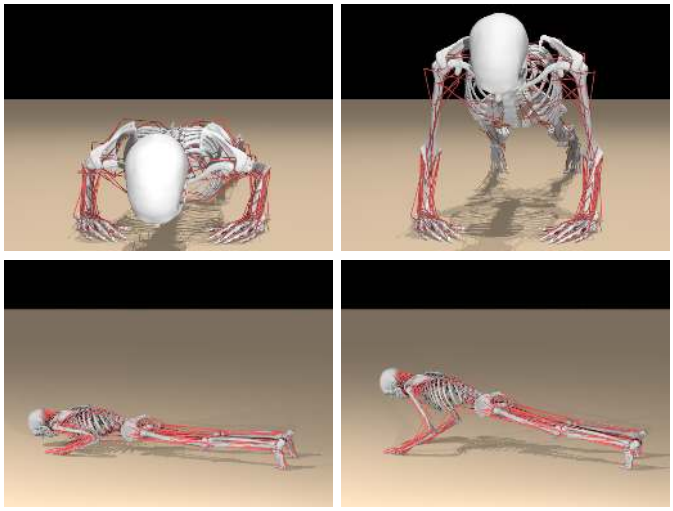

Fig. 11. **A skeleton performing push-ups demonstrates how our post-stabilization method can be used to generate muscular impulses to drive motion.**

and over 225 closed loops. The skeleton is controlled using key framed motion interpolated with B-splines. We achieved simple key framing by dragging bones with the mouse and applying post-stabilization. The skeleton uses global post-stabilization, while the iterative method from [22] is used on the net. Finally, we demonstrate our optimization-based inverse muscle actuation with an example of our skeleton performing a series of push-ups (Figure 11). The push-up motion was created using analytic functions, and our optimization method determined the muscle actuation required to achieve the desired joint angles. The skeleton contains muscles spanning multiple joints and antagonistic muscles, which our method handles trivially. Even with the additional calculation of the muscular impulses, this simulation only required three minutes a frame. Simple examples with lower degrees of freedom ran in interactive time.

Our method makes it easier to create realistic animations by making PD control easier to use, and that is a primary motivation of our work. Stiffness can be set on a per joint basis making the damping parameter k_p more intuitive. Although we could have incorporated motion capture and other methods into the motion descriptions to increase the realism of our examples, we chose not to and instead focused on highlighting the key points of our algorithm through simpler motions and poses.

VIII. Conclusion

We presented a novel approach to PD control that uses the analytic solution on a per joint basis coupled with inverse dynamics to alleviate difficulties with traditional PD control. Our method achieves the desired angle regardless of joint stiffness because it globally accounts for cross-coupling between joints and external forces such as gravity, effectively decoupling stiffness from control. We demonstrated the efficacy of our method in various scenarios including contact and collision, as well as on a complex articulated human skeleton with actuating muscles. For future work, we plan to apply our inverse muscle actuation framework to skinning (as proposed by [12]).

As mentioned previously, our method can be extended to work with more general controllers as long as one has a sense of an analytic or desired target trajectory. For example, more interesting human behaviors can be obtained with an approach similar to [5].

Appendix A

Pre-stabilization

Combining both position and angular constraints, we solve

$$\mathbf{f}(\mathbf{j},\mathbf{j}_\tau) = \begin{pmatrix} \mathbf{f}_t(\mathbf{j},\mathbf{j}_\tau) \\ \mathbf{f}_r(\mathbf{j},\mathbf{j}_\tau) \end{pmatrix} = \mathbf{0}$$

where $\mathbf{f}_t$ and $\mathbf{f}_r$ measure the translational and rotational joint errors after advancing forward in time. Using the notation of [22], these can be written as

$$\begin{aligned}\mathbf{f}_t(\mathbf{j},\mathbf{j}_\tau) = {} & \left(\mathbf{x}_p^w\right)^n + \Delta t \left(\mathbf{v}_p^w\right)^n + \frac{\Delta t}{m_p}\mathbf{j} \\ & + \hat{\mathbf{q}}\left(\triangle t \left(\omega_p^w\right)^n + \triangle t \mathbf{I}_p^{-1}\left(\mathbf{r}_p^* \mathbf{j} + \mathbf{j}_\tau\right)\right)\left[\left(\mathbf{q}_p^w\right)^n \left[\mathbf{x}_j^p + \mathbf{q}_j^p\left[\mathbf{x}^{target}\right]\right]\right] \\ & - \left(\mathbf{x}_c^w\right)^n - \Delta t \left(\mathbf{v}_c^w\right)^n + \frac{\Delta t}{m_c}\mathbf{j} \\ & - \hat{\mathbf{q}}\left(\triangle t \left(\omega_c^w\right)^n - \triangle t \mathbf{I}_c^{-1}\left(\mathbf{r}_c^* \mathbf{j} + \mathbf{j}_\tau\right)\right)\left[\left(\mathbf{q}_c^w\right)^n \left[\mathbf{x}_j^c\right]\right] = \mathbf{0}\end{aligned}$$

$$\begin{aligned}\mathbf{f}_r(\mathbf{j},\mathbf{j}_\tau) = {} & \hat{\mathbf{q}}\left(\triangle t \left(\omega_p^w\right)^n + \triangle t \mathbf{I}_p^{-1}\left(\mathbf{r}_p^* \mathbf{j} + \mathbf{j}_\tau\right)\right)\left(\mathbf{q}_p^w\right)^n \mathbf{q}_j^p \mathbf{q}^{target} \\ & - \hat{\mathbf{q}}\left(\triangle t \left(\omega_c^w\right)^n - \triangle t \mathbf{I}_c^{-1}\left(\mathbf{r}_c^* \mathbf{j} + \mathbf{j}_\tau\right)\right)\left(\mathbf{q}_c^w\right)^n \mathbf{q}_j^c = \mathbf{0}\end{aligned}$$

with $\mathbf{q}[\mathbf{r}]$ indicating a quaternion $\mathbf{q}$ applied to a vector $\mathbf{r}$. We solve using Newton iteration, and the Jacobian is a 7×6 matrix.

Acknowledgments

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ARO DAAD19-03-1-0331 and NIH U54-GM072970. R.W. was supported in part by a NIH NIGMS Fellowship.

References

[1] W. W. Armstrong and M. Green, "The dynamics of articulated rigid bodies for purposes of animation," in *Graph. Interface '85*, May 1985, pp. 407–415.

[2] J. Wilhelms and B. A. Barsky, "Using dynamic analysis to animate articulated bodies such as humans and robots," in *Graph. Interface '85*, May 1985, pp. 97–104.

[3] A. Witkin and M. Kass, "Spacetime constraints," in *Comput. Graph. (Proc. SIGGRAPH '88)*, vol. 22, 1988, pp. 159–168.

[4] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, "Animating human athletics," in *Proc. of SIGGRAPH '95*, 1995, pp. 71–78.

[5] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *ACM Trans. Graph. (SIGGRAPH Proc.)*, 2001, pp. 251–260.

[6] J. Wilhelms, "Virya–a motion control editor for kinematic and dynamic animation," in *Proc. on Graph. Interface '86/Vision Interface '86*, 1986, pp. 141–146.

[7] E. Kokkevis, D. Metaxas, and N. Badler, "User-controlled physics-based animation for articulated figures," in *Proc. Comput. Anim. '96*, 1996.

[8] M. Oshita and A. Makinouchi, "A dynamic motion control technique for human-like articulated figures," *Comput. Graph. Forum (Proc. Eurographics)*, vol. 20, no. 3, pp. 192–202, 2001.

[9] J. Laszlo, M. van de Panne, and E. Fiume, "Limit cycle control and its application to the animation of balancing and walking," in *Proc. of SIGGRAPH '96*, 1996, pp. 155–162.

[10] M. Neff and E. Fiume, "Modeling tension and relaxation for computer animation," in *Proc. ACM SIGGRAPH Symp. on Comput. Anim.*, 2002, pp. 77–80.

[11] J. Wilhelms, "Using dynamic analysis for realistic animation of articulated bodies," *IEEE Comput. Graph. and Appl.*, vol. 7, no. 6, pp. 12–27, 1987.

[12] P. Isaacs and M. Cohen, "Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics," in *Proc. of SIGGRAPH 1987*, 1987, pp. 215–224.

[13] P. Isaacs and M. Cohen, "Mixed methods for complex kinematic constraints in dynamic figure animation," *The Vis. Comput.*, vol. 4, no. 6, pp. 296–305, 1988.

[14] V. B. Zordan and J. K. Hodgins, "Tracking and modifying upper-body human motion data with dynamic simulation," in *In Proc. of Comput. Anim. and Sim. '99*, September 1999.

[15] V. B. Zordan and J. K. Hodgins, "Motion capture-driven simulations that hit and react," in *Proc. ACM SIGGRAPH Symp. on Comput. Anim.*, 2002, pp. 89–96.

[16] M. van de Panne, "Parameterized gait synthesis," *IEEE Comput. Graph. and Appl.*, vol. 16, no. 2, pp. 40–49, March 1996.

[17] M. McKenna and D. Zeltzer, "Dynamic simulation of a complex human figure model with low level behavior control," *Presence*, vol. 5, no. 4, pp. 431–456, 1996.

[18] B. Mirtich and J. Canny, "Impulse-based dynamic simulation," in *Alg. Found. of Robotics*, K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, Eds. A. K. Peters, Boston, MA, 1995, pp. 407–418.

[19] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proc. of 1995 Symp. on Int. 3D Graph.*, 1995, pp. 181–188, 217.

[20] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, no. 3, pp. 871–878, 2003.

[21] D. Kaufman, T. Edmunds, and D. Pai, "Fast frictional dynamics for rigid bodies," *ACM Trans. Graph. (SIGGRAPH Proc.)*, pp. 946–956, 2005.

[22] R. Weinstein, J. Teran, and R. Fedkiw, "Dynamic simulation of articulated rigid bodies with contact and collision," *IEEE Trans. on Vis. and Comput. Graph.*, vol. 12, no. 3, pp. 365–374, 2006.

[23] M. Cline and D. Pai, "Post-stabilization for rigid body simulation with contact and constraints." in *Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation, ICRA 2003*, Taipei, Taiwan, September 14-19 2003, pp. 3744–3751.

[24] W. Armstrong, M. Green, and R. Lake, "Near-real-time control of human figure models," *IEEE Comp. Graph. and Appl.*, vol. 7, no. 6, pp. 51–61, 1987.

[25] J. Wilhelms, M. Moore, and R. Skinner, "Dynamic animation: Interaction and control," *The Vis. Comput.*, vol. 4, pp. 283–295, 1988.

[26] S. Redon, N. Galoppo, and M. C. Lin, "Adaptive dynamics of articulated bodies," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 936–945, 2005.

[27] A. Bruderlin and T. W. Calvert, "Goal-directed, dynamic animation of human walking," in *Comp. Graph. (SIGGRAPH Proc.)*, 1989, pp. 233–242.

[28] M. McKenna and D. Zeltzer, "Dynamic simulation of autonomous legged locomotion," in *Comput. Graph. (Proc. SIGGRAPH '90)*, 1990, pp. 29–38.

[29] M. H. Raibert and J. K. Hodgins, "Animation of dynamic legged locomotion," in *Comp. Graph. (SIGGRAPH Proc.)*, 1991, pp. 349–358.

[30] M. van de Panne and E. Fiume, "Sensor-actuator networks," in *Proc. of SIGGRAPH '93*, 1993, pp. 335–342.

[31] M. van de Panne, R. Kim, and E. Fiume, "Virtual wind-up toys for animation," in *Proc. of Graph. Interface '94*, Banff, Alberta, Canada, 1994, pp. 208–215.

[32] H. Ko and N. I. Badler, "Animating human locomotion with inverse dynamics," *IEEE Comput. Graph. and Appl.*, vol. 16, no. 2, pp. 50–59, March 1996.

[33] J. Lo and D. Metaxas, "Recursive dynamics and optimal control techniques for human motion planning," in *Proc. of the Comput. Anim.*, 1999, p. 220.

[34] M. F. Cohen, "Interactive spacetime control for animation," in *Comp. Graph. (SIGGRAPH Proc.)*, 1992, pp. 293–302.

[35] Z. Liu, S. J. Gortler, and M. F. Cohen, "Hierarchical spacetime control," in *Proc. of SIGGRAPH '94*, 1994, pp. 35–42.

[36] Z. Popović and A. Witkin, "Physically based motion transformation," in *Comput. Graph. (Proc. SIGGRAPH '99)*, 1999, pp. 11–20.

[37] C. K. Liu, A. Hertzmann, and Z. Popović, "Learning physics-based motion style with nonlinear inverse optimization," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1071–1081, 2005.

[38] S. Delp and J. Loan, "A graphics based software system to develop and analyze models of musculoskeletal structures," *Comput. and Biomed. Research*, vol. 25, no. 1, pp. 21–34, 1995.

[39] S. Delp and J. Loan, "A computational framework for simulating and analyzing human and animal movement," *IEEE Computing In Science And Eng.*, vol. 2, no. 5, pp. 46–55, 2000.

[40] T. Komura, Y. Shinagawa, and T. Kunii, "A muscle-based feed-forward controller for the human body," *Comput. Graph. Forum (Proc. Eurographics)*, vol. 16, no. 3, pp. C165–C172, September 1997.

[41] T. Komura, Y. Shinagawa, and T. L. Kunii, "Calculation and visualization of the dynamic ability of the human body," *J. Vis. Comput. Anim.*, vol. 10, pp. 57–78, 1999.

[42] T. Komura, Y. Shinagawa, and T. L. Kunii, "Creating and retargetting motion by the musculoskeletal human body model," *The Vis. Comput.*, vol. 16, no. 5, pp. 254–270, June 2000.

[43] J. Teran, S. Blemker, V. Ng, and R. Fedkiw, "Finite volume methods for the simulation of skeletal muscle," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 68–74.

[44] J. Teran, E. Sifakis, S. Salinas-Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw, "Creating and simulating skeletal muscle from the visible human data set," *IEEE Trans. on Vis. and Comput. Graph.*, vol. 11, no. 3, pp. 317–328, 2005.

[45] I. Albrecht, J. Haber, and H. P. Seidel, "Construction and animation of anatomically based human hand models," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 98–109.

[46] W. Tsang, K. Singh, and E. Fiume, "Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion," in *Proc. ACM SIGGRAPH Symp. on Comput. Anim.*, 2005, pp. 319–328.

[47] S.-H. Lee and D. Terzopoulos, "Heads up! Biomechanical modeling and neuromuscular control of the neck," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 25, no. 3, pp. 1188–1198, 2006.

[48] T. Komura and Y. Shinagawa, "Motion conversion based on the musculoskeletal system," in *Proc. of Graph. Interface 2001*, June 2001, pp. 27–36.

[49] V. De Sapio, J. Warren, O. Khatib, and S. Delp, "Simulating the task-level control of human motion: a methodology and framework for implementation," *The Vis. Comput.*, vol. 21, no. 5, pp. 289–302, June 2005.

[50] D. K. Pai, S. Sueda, and Q. Wei, "Fast physically based musculoskeletal simulation," in *SIGGRAPH 2005 Sketches & Appl.* ACM Press, 2005.

[51] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Comput. Meth. in Appl. Mech. and Eng.*, vol. 1, pp. 1–16, 1972.

[52] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 594–603, 2002.

[53] B. Garner and M. Pandy, "Musculoskeletal model of the upper limb based on the visible human male dataset," *Comput. Meth. Biomech. Biomed. Eng.*, vol. 4, pp. 93–126, 2001.

[54] S. Delp, J. Loan, M. Hoy, F. Zajac, E. Topp, and J. Rosen, "An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures," *IEEE Trans. on Biomed. Eng.*, vol. 37, pp. 757–767, 1990.

[55] K. Holzbaur, W. Murray, and S. Delp, "A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control," *Annals of Biomed. Eng.*, vol. 33, no. 6, pp. 829–840, 2005.