Motion Perturbation Based on Simple Neuromotor Control Models

Michael B. Cline, KangKang Yin and Dinesh K. Pai
University of British Columbia
March 30, 2002

Abstract

Motion capture is widely used for character animation. One of the major challenges in this area is modifying human motion in plausible ways. Previous work has focused on transformations based on kinematics and dynamics, but has not explicitly taken into account the emerging knowledge of how humans control their motion. In this paper we show how this can be done using a simple human neuromuscular control model. Our model of muscle forces includes a feedforward term and low gain passive feedback. The feedforward component is calculated from motion capture data using inverse dynamics. The feedback component generates reaction forces to unexpected external disturbances. The perturbed animation is then resynthesized using forward dynamics. This allows us to create animations where the character reacts to unexpected external forces in a natural way (e.g.,when the character is hit by a flying object), but still retains qualities of the original animation. Such technique is useful for applications such as interactive sports video games.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: physically-based simulation, motor control, motion capture

1 Introduction

Traditional keyframing animation, motion capture, and physical simulation are the three major computer animation techniques widely used today. Among the three techniques, motion capture is increasing in popularity for realistic and stylistic human-like figure animation. However, the amount of motion we can capture still cannot meet our needs. In applications such as sports video games, the lack of variation in similar motions or the lack of changes due to novel situations greatly reduces the sense of reality.

There has been considerable research in automatic motion editing and transformation techniques. We review related work in Section 2. Kinematic motion editing is relatively cheap, but ignores dynamic constraints. Dynamic motion transformation can guarantee physical plausibility, and interactions between characters and their environment can be handled naturally.

However, dynamic human simulation still suffers from being unrealistic. We believe that the lack of realism is primarily due to the lack of human motion control models and not due to inadequacies in physical modeling. After all, an industrial robot obeys the same laws of physics as humans and also moves in a physically plausible way. But its motion does not look human because its motors and control algorithms are different from those of humans.

Dynamic simulation is also expensive, not because of the cost of dynamics computations (which is reducing rapidly due to Moore's law), but due to the difficulty of creating realistic dynamic models and controllers. One approach to handling this motor control problem is to manually design motor controllers for various motor skills. This turns out to be even harder than keyframing animation, and often results in robotic motions. Another approach is to use optimal control theory to solve for an optimal motion, given some empirical objective functions. This also results in loss of stylistic details, and the computational cost is extremely high. Thus it would be useful if at least a part of the control can be estimated from motion capture data.

This situation suggests we explicitly take into account human motor control mechanisms for human character simulation. One way is to directly learn motor control mechanisms from the motion capture data. Motion capture data encapsulates much knowledge of how humans control their movements, and also contains rich style information. The other possibility is to borrow research in human motor control from neuroscience, biomechanics and other related movement sciences. This is still an active and contentious research topic with its own arena, and the mystery of human movement control is far from being completely revealed. Nevertheless, even simplified models and general principles of human motor control can be useful in increasing the realism of computer animation.

This paper proposes a way to incorporate a simple human neuromuscular control model into dynamic simulation systems. Original motion capture animations can be modified adaptively according to small unexpected disturbances rising from a dynamically changing environment. Biological noise can also be introduced on purpose to avoid repetitive motions.

We will first review some related work in Section 2, and then describe our motor control model in Section 3. In Section 4 we give the relevant details of our dynamic simulation system. In Section 5 we describe how the dynamic system is coupled with the proposed motor controller, i.e., how the feedforward torques are estimated from motion capture data, and how the feedback torques are computed during the forward simulation. Experimental results are shown in Section 6 with relevant discussions. We conclude with Section 7 and point out possible future work.

2 Related Work

Several researchers have approached the motion editing problem from a purely kinematic, signal processing point of view [Bruderlin and Williams 1995; Witkin and Popovic 1995; Unuma et al. 1995; Lee and Shin 1999]. Various signal processing techniques, including multiresolution filtering, displacement mapping, interpolation,

extrapolation, warping and blending are applied to kinematic motion data.

Highly skilled specialists have successfully designed motor controllers for dynamic human simulation by hand [Hodgins et al. 1995]. Such controllers are composable using machine learning techniques [Faloutsos et al. 2001]. Incoporating motion capture into dynamic simulations makes the control problem much easier to solve, which in its simplest form is to directly use a tracking controller connected to a motion capture device [Zordan and Hodgins 1999]. Adjusting controller parameters results in force-based editing techniques [Pollard and Behmaram-Mosavat 2000].

Spacetime constraints (SC), first proposed by Witkin and Kass [Witkin and Kass 1988], put the motion editing problem into a constrained optimization framework. SC combines kinematic keyframimg (space constraints) with dynamic simulation (time constraints). Conceptually, a constrained optimizer is used as the motor controller. The optimization approach has also been used and is still in use in relevant neuroscience and movement science disciplines. There are various improvements and extensions to the basic SC approach [Cohen 1992; Ngo and Marks 1993; Liu et al. 1994; Rose et al. 1996; Gleicher 1998; Popovic and Witkin 1999]. They either try to make SC practical by addressing the inherent high cost of the optimization, or extend SC to address specific types of animation tasks. Muscle dynamics can also be incorporated into the SC framework[Komura and Shinagawa 1997; Komura et al. 2001].

Motor learning techniques [van de Panne and Fiume 1993; Grzeszczuk and Terzopoulos 1995; Grzeszczuk et al. 1998] for physically-based animation are able to discover motor controllers for basic motion tasks such as locomotion. Body configuration and evolution/optimization criteria are provided by the user. It is unclear how one could introduce explicit motion control knowledge into this framework, and how to scale this technique to more complex systems like entire humans.

The related work from neuroscience and movement science will be described in detail in the following section.

3 A Biologically-based Motor Control Model

The computational study of biological motor control is fundamentally concerned with the relationship between sensory signals and motor commands [Jordan and Wolpert 1999; Wolpert and Ghahramani 2000]. There are two basic transformations involved. The first transformation is from sensory signals to motor commands. It is often refered to as motion planning in Computer Graphics community. Our central nervous system (CNS) issues the motor commands for a specific task based on *internal models* we have learned. The second transformation is from motor commands to their sensory consequences. It is governed by the physics of the environment, the musculoskeletal system and sensory receptors.

One fundamental fact of biological motor systems is that neurons, and especially the chemical synapses between them, are very slow. Therefore sensory feedback through the periphery is delayed by a significant amount. For example, visual feedback on arm movements ranges from 150-250ms. Even a spinal reflex loop involving as few as three neurons can take on the order of 30ms. These are very large delays when compared with the total movement duration of very fast (150ms) to intermediate (500ms) movements. Such delays can result in instability when trying to make rapid movements under high-gain feedback control. Therefore, high-gain feedback controllers which are widely used in robotics and control engineering are unrealistic for biological systems.

During the last decade, it has become increasingly accepted that the brain utilizes internal models of dynamics in planning and controlling motion. The internal model theory proposes that the brain needs to acquire an inverse dynamics model of the object to be controlled through motor learning, after which motor control can be executed using feedforward muscle forces, in an almost open-loop manner [Kawato 1999; Mussa-Ivaldi 1999]. This explains why our movements show highly stereotyped and stylized patterns, although almost any task can in principle be achieved in infinitely many different ways. This also explains the observation that well-trained movements exhibit relatively low joint stiffness (i.e., changes in net torque at the joint due to displacements from the reference motion), while during motor learning the stiffness is higher due to lack of good internal models.

The internal models consist of two modules: *forward models* and *inverse models*. Forward models predict the behavior of the body and world with its knowledge about the body dynamics and environment. Inverse models invert the system by providing the motor command which will cause a desired change in state. Forward models are important in motor planning in biological systems, since they can provide fast internal predictive feedback instead of relying only on the delayed feedback from the periphery. How these models are used by the brain for planning motion is currently being investigated [Harris and Wolpert 1998].

The precise details of the internal models are not necessary to transform human motion due to unexpected disturbances of short duration. It is sufficient to know the important role of feedforward torques in generating human motion, and the use of low gain feedback. Instead of computing the feedforward torques, we directly estimate it for a given motion using motion capture data as shown in Section 5.

The intrinsic mechanical properties of muscles and tendons produce proportional (stiffness) and derivative (viscosity) feedback forces without delay [Hall 1998]. Our model uses this muscle property as a low-gain feedback controller to stabilize the limb along the desired trajectory. The muscle force-length relationships can be quite complex, but it is well known that muscle stiffness increases with generated force [Wise and Shadmehr 2002]. A simple model of this non-linear relationship is the so-called bilinear model of muscle impedance [Hogan 1990; Winters and Crago 2000]. This model implies that the effective stiffness is proportional to the neural input, and hence the generated muscle forces. We assume that muscle viscoelasticities also increase in a similar way and are small for well-trained movements.

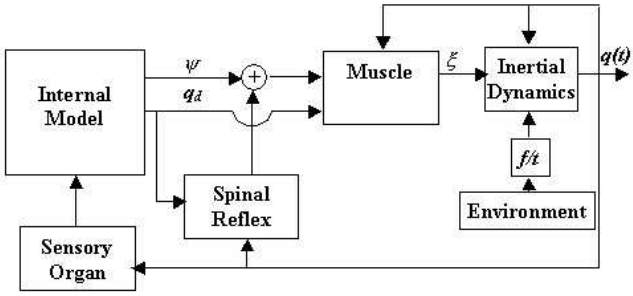

Figure 1: Motor control model for motion transformation upon unexpected external disturbances

Figure 1 shows our reference motor control model [Wang et al. 2001]. The internal model is treated as a black box whose output is the feedforward motor command ψ and desired trajectory q_d. The muscle-tendon system is driven by the motor command and generate the force ξ. Muscle force and external force act upon the human inertial dynamic system and produce the actual trajectory. There are three feedback paths: muscle-tendon feedback which has

essentially zero delay; spinal reflex which has 30-50ms delay; and sensory feedback to the brain which has 150-250ms delay.

Our assumption is that for short duration unexpected disturbances such as being hit by a ball, the brain has no time to complete the long latency feedback loops and replan the motion. The trajectory is restored by the low gain muscle-tendon feedback forces. Thus we simplify our model by omitting the long latency feedback modules and only consider the muscle feedback module. The muscle feedback controller we use is a hard-wired, low-gain and signal-dependent feedback controller. For well-trained unperturbed motion, muscle feedback has low gain. So we can estimate the muscle force ξ by inverse dynamics from motion capture data, and use ξ as an approximation of feedforward command ψ (see Section 5.1).

4 Dynamic Simulation of Motion Capture Data

We have developed a general-purpose rigid body simulation system, and extended it to meet the requirements of motor control. Our simulator is based on a Lagrange multiplier approach for computing constraint forces, inspired by the work of Baraff [Baraff 1996] (see references of Baraff for other work in the area). We extend this approach by allowing the simulator to solve both forward and inverse dynamics problems. We also introduce some new matrix notation that we use to describe the muscle forces and include them in our equations of motion.

4.1 Dynamics Framework

In the Lagrange multiplier approach, the velocity of each body is parameterized by a full 6 coordinate representation. Each joint in an articulated body is represented by a constraint equation, which is a linear equation on the velocities of the bodies. Constraint i between bodies a and b is given by the equation:

$$\mathsf{j}_{ia}\mathbf{v}_a + \mathsf{j}_{ib}\mathbf{v}_b = \mathbf{0}, \tag{1}$$

where the Jacobian matrices j have 6 columns and one row for each degree of freedom they remove from the system (the number of rows is referred to as the degree of the constraint). For a system with many constraints and many bodies we construct one large jacobian matrix J, containing all of the constraint equations, and concatenate the velocities of all of the bodies into a single vector $\mathbf{v}$. For example, if we had a chain of four rigid bodies connected by joints, the constraint equation would appear as follows:

$$\mathsf{J}\mathbf{v} = \begin{bmatrix} \mathsf{j}_{1a} & \mathsf{j}_{1b} & 0 & 0 \\ 0 & \mathsf{j}_{1b} & \mathsf{j}_{1c} & 0 \\ 0 & 0 & \mathsf{j}_{1c} & \mathsf{j}_{1d} \end{bmatrix} \begin{bmatrix} \mathbf{v}_a \\ \mathbf{v}_b \\ \mathbf{v}_c \\ \mathbf{v}_d \end{bmatrix} = \mathbf{0} \tag{2}$$

If the constraint is workless (i.e., a frictionless joint), then the constraint forces are multiples of the rows of J. The sum of all constraint forces is given by

$$\mathbf{f}_c = \mathsf{J}^T\lambda, \tag{3}$$

where λ is a vector of Lagrange multipliers. The dimension of λ is the sum of the degrees of all of the constraints, which we will denote with n.

The row space of J is the space of constraint forces. We now wish to introduce an analogous matrix H whose row space is the space of all possible "muscle forces" which the joints can apply to their neighboring bodies. The rows of H correspond to equal and opposite torques applied at the joint. Similar to equation 3, the muscle forces are given by

$$\mathbf{f}_m = \mathsf{H}^T\tau. \tag{4}$$

Combining the constraint equations with the Newton-Euler equations of motion gives us the following matrix equation:

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T & -\mathsf{H}^T \\ \mathsf{J} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \lambda \\ \tau \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{ext} \\ 0 \end{bmatrix} \tag{5}$$

Here M is the mass-inertia matrix of the bodies in the system (a block diagonal matrix with each block corresponding to one body), $\mathbf{a}$ is the acceleration vector of the bodies, and vector $\mathbf{f}_{ext}$ contains external forces such as gravity and coriolis forces.

Equation 5 is a unified expression that is true for both forward and inverse dynamics. We will later rearrange this equation to reflect the known and unknown values in these two types of problems.

4.2 Details of Matrices J and H

For a joint connecting one body to another, the matrices J and H both have twelve columns (they multiply with a vector $[v_a^T\ \omega_a^T\ v_b^T\ \omega_b^T]^T$ containing the linear and angular velocities of both bodies that the joint is connected to). The number of rows in J is the number of degrees of freedom (DOF) that the joint removes from the system, while the number of rows of H is the number of degrees of freedom in the rotation of the joint. These two numbers always add up to six. For instance, in a hinge joint, there is 1 degree of rotation freedom, and five DOF are removed from the system. For a ball joint, there are 3 degrees of rotational freedom, and 3 DOF are removed. If we recall that the rows of these matrices are used as force basis vectors, we can also note that the first and second halves of these vectors must correspond to equal and opposite forces applied to the pair of bodies. This constraint restricts the row space of J and H to a six-dimensional subspace of $\mathbb{R}^{12}$. The row space J and H must always span this entire 6D subspace containing all equal-and-opposite force pairs.

In our implementation, we follow the convention for describing rigid body velocities that is described by Baraff and Witkin [Baraff and Witkin 1997]. The velocity of a rigid body is given as a vector $\mathbf{v} = [v^T\ \omega^T]^T$, where v^T is the velocity of the centre of mass of the object, given in world coordinates, and ω is the world coordinates of the angular velocity vector. Under this convention, our matrices J and H has the following structure:

$$\begin{bmatrix} \mathsf{J} \\ \mathsf{H} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & & -1 & 0 & 0 & \\ 0 & 1 & 0 & -[\mathbf{r}_a] & 0 & -1 & 0 & [\mathbf{r}_b] \\ 0 & 0 & 1 & & 0 & 0 & -1 & \\ 0 & 0 & 0 & e_1^T & 0 & 0 & 0 & -e_1^T \\ 0 & 0 & 0 & e_2^T & 0 & 0 & 0 & -e_2^T \\ 0 & 0 & 0 & e_3^T & 0 & 0 & 0 & -e_3^T \end{bmatrix}, \tag{6}$$

where $\mathbf{r}_a$ is the vector from the centre of mass of body A to the joint, and $\mathbf{r}_b$ is the vector from body B's centre of mass to the joint. The notation $[\mathbf{r}]$ denotes the 3×3 skew-symmetric cross product matrix of vector $\mathbf{r}$. The axis vectors e_1, e_2, e_3 are three orthogonal vectors in world coordinates, some of which are the free rotation axes of the joint, and some of which may be axes that joint bodies are constrained not to rotate around. Which of the rows of the above matrix belong to J and which belong to H depends on the type of joint.

For a more detailed derivation, see [Cline 2002].

4.3 Forward Dynamics

In forward dynamics, the muscle force multipliers τ are known quantities. Moving $\mathsf{H}^T\tau$ to the right hand side of equation 5, and then discretizing gives

$$\begin{bmatrix} \mathsf{M} & -\mathsf{J}^T \\ \mathsf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathsf{M}\mathbf{v}_t - h\mathbf{k} \\ 0 \end{bmatrix} \tag{7}$$

where h is the time step size, $\mathbf{v}_t$ and $\mathbf{v}_{t+h}$ are the velocity of the rigid bodies at time t and $t+h$, and $\mathbf{k} = \mathbf{f}_{ext} + \mathsf{H}^T\tau$. Solving this equation at each time step gives us the updated velocity of the system.

To counteract drift at the joints due to numerical error, we use a post-step stabilization scheme [Ascher et al. 1995; Cline 2002], where after each simulation step we make a small correction to the position of the bodies so that the constraints are maintained.

4.3.1 Accomodating Stiffness

When using explicit integrators for forward simulation, we found that for the simulation to remain stable, we were forced to use small time steps and unrealistically low stiffness and damping constants, making our model very sensitive to external forces. In order for the techniques in this paper to work well, an implicit integrator is essential.

We use an implementation of the linearly implicit time stepping method [Anitescu and Potra 2000]. Fortunately this requires only small modifications to our existing simulator. The main requirement of this implicit method is that we must calculate the gradients of the stiff forces with respect to changes in the position and velocity of the rigid bodies.

To derive the linearly implicit method, we start with a backward Euler discretization of rigid body dynamics equation:

$$\mathsf{M}\left(\frac{\mathbf{v}_{t+h} - \mathbf{v}_t}{h}\right) = \mathbf{k}(\mathbf{p}_{t+h}, \mathbf{v}_{t+h}, t+h) \tag{8}$$

The difficulty in solving this is that the force vector $\mathbf{k}$ is not known unless the position and velocity vectors $\mathbf{p}_{t+h}$ and $\mathbf{v}_{t+h}$ are known. We make the linear approximation (hence the term 'linearly implicit') that

$$\begin{aligned}&\mathbf{k}(\mathbf{p}_{t+h}, \mathbf{v}_{t+h}, t+h) \approx \\ &\quad \mathbf{k}(\mathbf{p}_t, \mathbf{v}_t, t) + \nabla_p\, h\, \mathbf{v}_{t+h} + \nabla_v(\mathbf{v}_{t+h} - \mathbf{v}_t),\end{aligned} \tag{9}$$

where ∇_p and ∇_v are the gradients of the function $\mathbf{k}$ with respect to change in position and velocity, respectively, and evaluated at $(\mathbf{p}_t, \mathbf{v}_t, t)$.

If we substitute this into equation 8 and move all of the terms with $\mathbf{v}_{t+h}$ to the left hand side, we obtain

$$\begin{aligned}&(\mathsf{M} - h^2\nabla_q - h\nabla_v)\mathbf{v}_{t+h} = \\ &\quad \mathsf{M}\mathbf{v}_t - h\nabla_v\mathbf{v}_t + h\,\mathbf{k}(\mathbf{p}_t, \mathbf{v}_t, t)\end{aligned} \tag{10}$$

It is convenient to use the notation

$$\hat{\mathsf{M}} = \mathsf{M} - h^2\nabla_q - h\nabla_v \tag{11}$$

and

$$\hat{\mathbf{k}} = \mathbf{k}(\mathbf{p}_t, \mathbf{v}_t, t) - \nabla_v\mathbf{v}_t \tag{12}$$

so that the linearly implicit equation for forward dynamics closely resembles equation 7. The implicit version is:

$$\begin{bmatrix} \hat{\mathsf{M}} & -\mathsf{J}^T \\ \mathsf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathsf{M}\mathbf{v}_t - h\hat{\mathbf{k}} \\ 0 \end{bmatrix} \tag{13}$$

In our implementation, we estimate the force gradients ∇_p and ∇_v numerically, by evaluating $\mathbf{k}$ for several position and velocity values in the neighborhood of the current state of the system [Anitescu and Potra 2000].

4.4 Inverse Dynamics

Inverse dynamics is the process of finding a set forces that explain a given motion. The inverse dynamics equations we solve are another form of equation 5. We move $\mathsf{M}\mathbf{a}$ to the right hand side of the equation (because the acceleration is a known quantity). Assuming the constraint equations $\mathsf{J}\mathbf{v} = \mathbf{0}$ are satisfied by the given motion, we no longer need the second row of equation 5. We are left with:

$$\begin{bmatrix} \mathsf{J}^T & \mathsf{H}^T \end{bmatrix} \begin{bmatrix} \lambda \\ \tau \end{bmatrix} = \mathsf{M}\mathbf{a} + \mathbf{f}_{ext} \tag{14}$$

If we can estimate the mass properties of the bodies of our articulated figure, along with the accelerations of its component bodies, then equation 14 can be solved to determine the muscle forces multipliers τ.

5 Implementation Details

The implementation of our method consists of two main components. The first is a preprocessing stage, where we use inverse dynamics to estimate the feedforward torques from the motion capture data. The second component, which happens during the dynamic simulation, is the calculation of the actual muscle torques, which are a combination of the precomputed feedforward torques, and feedback torques which depend on the difference between the trajectories of the rigid bodies in the motion capture and the trajectories in the dynamic simulation.

5.1 Inverse Dynamics Preprocessing

Before beginning our dynamic simulation, we estimate a set of feedforward muscle torque multipliers τ for each time step. Given the mass matrix M, the constraint jacobian J, the matrix H, the external force vector $\mathbf{f}_{ext}$, and the acceleration $\mathbf{a}$, we can calculate τ using equation 14.

We can estimate the accelerations of the rigid bodies in each frame by fitting a smooth curve to the position data, and then finding the second derivative of the curve.

One difficulty in evaluating equation 14 is that the mass properties of the character's component rigid bodies are unknown. We deal with this by approximating the shape of the character with polyhedra and computing the mass matrix for these, assuming the density of water (the body's average density is reasonably close to that of water).

We believe it may be possible to directly estimate the mass-inertia matrix of the figure from the motion capture data by making the mass properties unknowns in the inverse dynamics equation. However, this approach would require solving the inverse dynamics for every frame in the animation simultaneously, making it more computationally challenging. Estimating the mass properties from the geometry, on the other hand, allows us to solve each frame's inverse dynamics separately.

We can use equation 6 to compute J and H. In order to do this, we require a kinematic model of the character to tell us the location and type of each joint. In our experiments, this model was given with the data set.

Using equation 14, we precompute feedforward torques for each of the frames in the animation before beginning the forward simulation. The total feedforward torque is given by $\mathsf{H}^T\tau$. But for the next section, we will need to break this down into smaller components $\psi_1, ..., \psi_n$, each of which correspond to one degree of freedom of the joints.

$$\begin{aligned} \mathsf{H}^T\tau &= \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \dots & \mathbf{h}_n \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} \\ &= \mathbf{h_1}\tau_1 + \mathbf{h_2}\tau_2 + \ldots + \mathbf{h_n}\tau_n \\ &= \psi_1 + \psi_2 + \ldots + \psi_n \end{aligned} \tag{15}$$

The ψ's are stored in body relative coordinates rather than world coordinates, because the orientation of the joints with respect to the world may be different in forward simulation than in the original motion capture animation.

5.2 Combining Feedback and Feedforward Torques During Forward Simulation

The muscle torques applied during forward simulation are a combination of the feedforward torque (the torques calculated during the initial inverse dynamics phase) and the feedback torque, which models the muscle dynamics module of our motor control model.

Individual feedback torques are calculated for each degree of freedom of all of the joints. Let $\theta_1, \theta_2, \ldots \theta_n$ be joint angles corresponding to each degree of freedom, and $\dot{\theta}_1, \dot{\theta}_2, \ldots \dot{\theta}_n$ be joint velocities. The joint angles $\theta_{d1}, \theta_{d2}, \ldots \theta_{dn}$ are the "desired" joint angles – the joint angles from the motion capture data.

The feedback torque tries to compensate small drifts and disturbances during the simulation. It is given by

$$\gamma_i = \mathbf{h}_i |\tau_i| \left(k_s(\theta_{di} - \theta_i) + k_d(\dot{\theta}_{di} - \dot{\theta}_i) \right) \tag{16}$$

where k_s and k_d are the stiffness and damping constants. Note that the stiffness is proportional to the magnitude of the muscle torque, $\mathbf{h}_i|\tau_i|$, as observed empirically in muscle biomechanics.

Our total muscle torques are given by the sum of the feedforward and feedback torques:

$$\xi = \sum_{i=1}^{n} (\gamma_i + \psi_i) \tag{17}$$

The muscle torques are added into the dynamics equation along with any other external forces, such as gravity and perturbations.

5.3 Algorithm Summary

Our approach can be summarized by the following steps:

- Preprocessing: inverse dynamics
  - Estimate the mass matrix M for rigid bodies which approximate the shape of the character.
  - Fit a smooth curve to the position data.
  - Sample the accelerations of the rigid bodies at the rate at which we wish to run the dynamic simulation.
  - For each sampled time step:
    - Compute J and H given the positions of the rigid bodies.
    - Solve the inverse dynamics equation to determine the muscle torque multipliers τ.
    - Store the feedforward torques $\psi_1, \ldots, \psi_n$ in body coordinates.
    - Store the current joint angles θ_d and joint velocities $\dot{\theta}_d$.
- For each step during forward simulation:
  - Compute the current joint angles θ and joint velocities $\dot{\theta}$.
  - Compute the feedback torques $\gamma_1, \ldots, \gamma_n$, using equation 16.
  - Compute the total external force $\mathbf{f}_{ext}$.
  - Solve the dynamics equation 13 to determine the state of the system at the next time step.

6 Experiments and Discussions

We perform our experiments on captured arm motions and full-body motions in football games. Figure 2 shows the skeleton model and the surface model we use in our simulation and rendering. Although our skeleton model is relatively complex for the dynamic system, it is still very simple for realistic human motion simulation. For example, the whole spine is represented by only 3 segments, which will cause unrealistic body response under some cases. For people who are familiar with human anatomy, the human shoulder is far from an ideal ball-and-socket joint. The supination and pronation of hand is achieved by the proximal and distal radioulnar joints in real life, not by an 1dof joint in-between the elbow and the wrist. The surface geometry is of rather low resolution, 590 vertices for the body and 136 vertices for the helmet, which may also degrade realism to some extent.

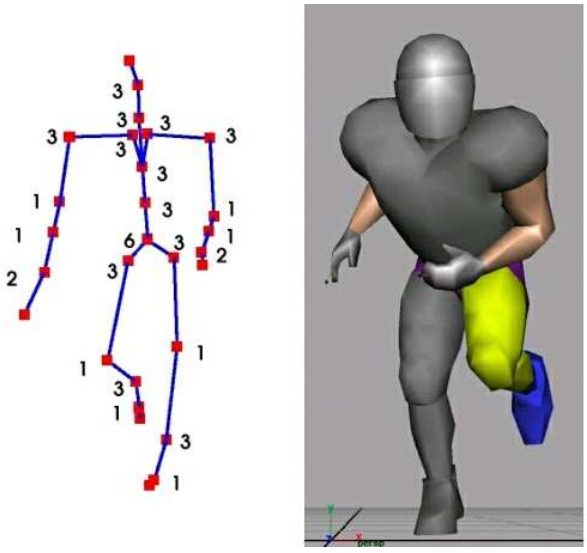

Figure 2: The skeleton model for our character simulation is shown on the left. Each square represents a joint. The total degrees of freedom is 54, and the degrees of freedom for each joint are labeled nearby. The skin geometry model for final rendering is shown on the right.

Figure 3 shows motion perturbation on a sequence of arm motion. Figure 4 shows motion perturbation on a sequence of full-body motion. In both cases, the simulated skeleton responds to external disturbances and restores to the original motion naturally. Skinned character animations are shown on the video that accompanies this paper.

There are several limitations in this work. First, our motor control model is very simple right now. It can only cope with small disturbances. We work under the assumption that small disturbances

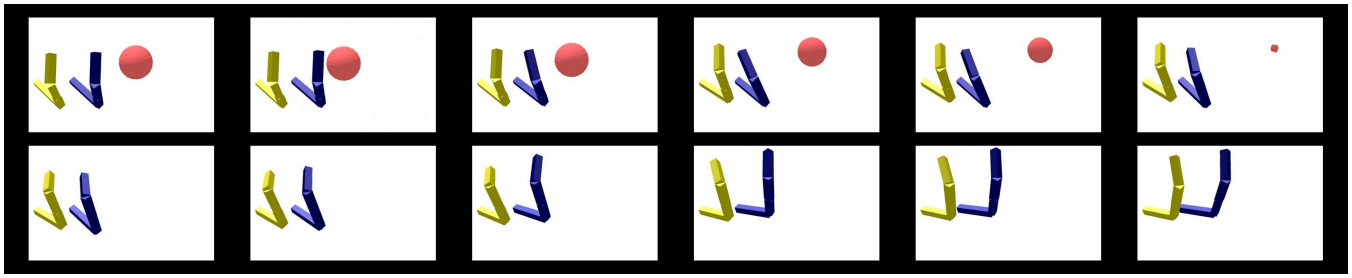

Figure 3: A sequence of a ball hitting a simulated arm (left to right, top to bottom). The motion capture data (left arm in each frame) is shown for comparison. The simulated arm reacts to the impact of the ball (which happens between frames 2 and 3), and then returns to the same path as the motion capture.

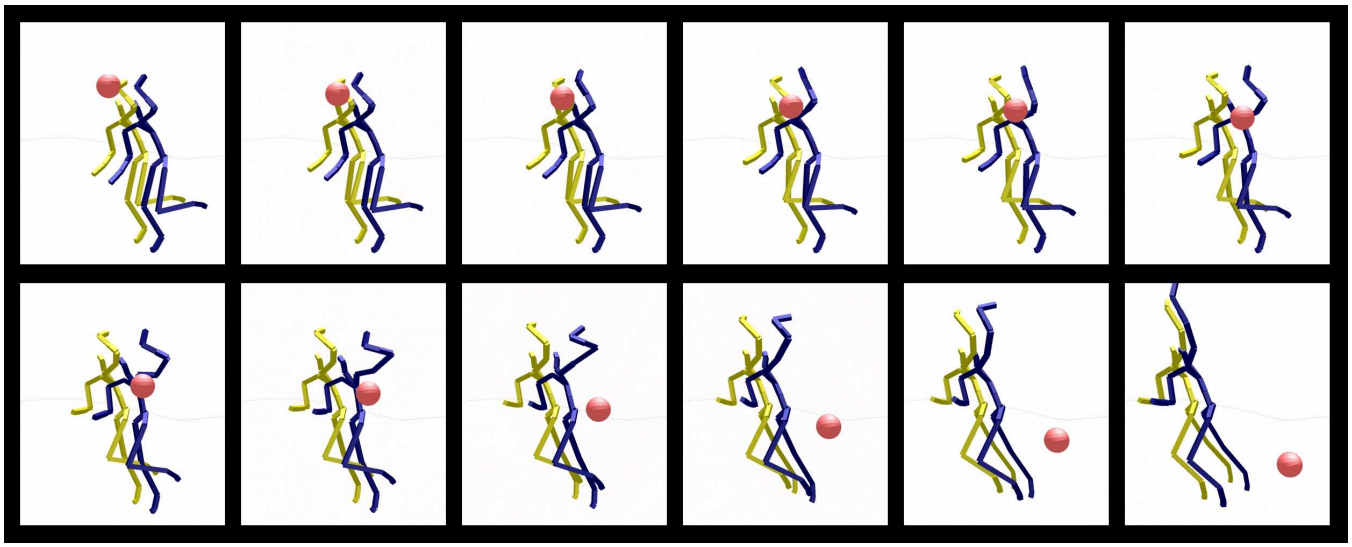

Figure 4: A sequence of a ball hitting a simulated body (left to right, top to bottom). The motion capture data (left body in each frame) is shown for comparison. The simulated body reacts to the impact of the ball (two collisions occur, one between frames 2 and 3, the other between frames 5 and 6), and then restores to the original motion.

are totally recoverable by muscles, without triggering the brain to replan the motion. In a real-life interactive sports video game, realistic dodges and realistic falls are absolutely desirable but very difficult. We know of no system that can do this yet. More sophisticated motor control models need to be developed. We also simplify the system by constraining the root joint (the joint located roughly at the Lumbosacral angle of the spine) to move along the motion capture path. We can thus omit the contact dynamics with the floor. This turns out acceptable since we are dealing with small upper limb perturbations. Even though motion perturbation is a subset of possible motion modifications, it is a large and common subset important for video games.

We developed our own dynamic simulator instead of using a commercial package. We simulate dynamics in maximal coordinates instead of in reduced (generalized) coordinates. These decisions make our work different from most of other works on human simulation in Computer Graphics community, however, they give us better flexibility. These decisions do bring in problems. Currently the performance is not real time. For a full body with an extra ball, the simulation runs at 7 frames per second on a Pentium III machine. The frame rate drops even further when collision happens. By simple performance analysis, we found out the computation mainly goes to: finite-differencing for the implicit integrator (see Section 4.3.1), forward dynamics, and post-step stabilization (both see Section 4.3). The finite-differencing can be sped up by distributed parallel computation, or better yet, replaced by analytical differentiation or their approximations. For forward dynamics, we tried the linear-time approach [Baraff 1996] without auxiliary constraints. It only gives us about a factor of two speedup for a full body system with 84 Langrange multipliers. This conforms to Baraff's results [Baraff 1996], where a system with 99 Lagrange multipliers has a factor of two speedup. For more complex systems, such as several-player simulations, linear-time dynamics is definitely the right way to go.

7 Conclusion

Motor control is one of the major challenges in physically based human simulation. In this paper, we address this problem by explicitly incorporating human neuromotor control models into the human simulation system. We test our approach with motion perturbation tasks on motion capture data, and the results are promising. We believe using a biologically-based motor control module is the ultimate way to solve control problems in physically-based character animation. We intend to explore better computational motor control models from the biomechanics and neural control literature [Winters and Crago 2000; Wise and Shadmehr 2002], and apply them to more challenging motion transformation tasks.

Acknowledgements

The authors would like to thank Electronic Arts for providing us motion capture data for this research.

References

ANITESCU, M., AND POTRA, F. A. 2000. A time-stepping method for stiff multibody dynamics with contact and friction. *Reports on computational mathematics, ANL/MCS-P884-0501, Mathematics and Computer Science division, Argonne National Laboratory*.

ASCHER, U. M., CHIN, H., , PETZOLD, L. R., AND REICH, S. 1995. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Mechanics of Structures and Machines 23*, 135–158.

BARAFF, D., AND WITKIN, A. 1997. *Physically Based Modeling: Principles and Practice*. Siggraph '97 Course notes.

BARAFF, D. 1996. Linear-time dynamics using lagrange multipliers. In *Proceedings of SIGGRAPH 1996*, ACM Press / ACM SIGGRAPH, H. Rushmeier, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 137–146.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, R. Cook, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 97–104.

CLINE, M. B. 2002. *To Be Titled*. Master's thesis, University of British Columbia.

COHEN, M. F. 1992. Interactive spacetime control for animation. In *Proceedings of SIGGRAPH 1992*, ACM Press / ACM SIGGRAPH, E. E. Catmull, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 293–302.

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

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, M. Cohen, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 33–42.

GRZESZCZUK, R., AND TERZOPOULOS, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, R. Cook, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 63–70.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, M. Cohen, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 9–20.

HALL, S. J. 1998. *Computer Facial Animation*, second ed. Mosby.

HARRIS, C. M., AND WOLPERT, D. M. 1998. Signal-dependent noise determines motor planning. *Nature*, 394, 780–784.

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

HOGAN, N. 1990. *Mechanical Impedance of Single and Multi-Articular Systems*. Springer-Verlag.

JORDAN, M. I., AND WOLPERT, D. M. 1999. *Computational motor control*. MIT Press.

KAWATO, M. 1999. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology 9*, 718–727.

KOMURA, T., AND SHINAGAWA, Y. 1997. A muscle-based feed-forward controller of the human body. In *Computer Graphics Forum (Proceedings of Eurographics 1997)*, 165–176.

KOMURA, T., SHINAGAWA, Y., AND KUNII, T. L. 2001. Attaching physiological effects to motion-captured data. In *Graphics Interface Proceedings*, 27–36.

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

LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical spacetime control. In *Proceedings of SIGGRAPH 1994*, ACM Press / ACM SIGGRAPH, A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 24–29.

MUSSA-IVALDI, F. A. 1999. Modular features of motor control and learning. *Current Opinion in Neurobiology 9*, 713–717.

NGO, J. T., AND MARKS, J. 1993. Spacetime constraints revisited. In *Proceedings of SIGGRAPH 1993*, ACM Press / ACM SIGGRAPH, J. T. Kajiya, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 343–350.

POLLARD, N. S., AND BEHMARAM-MOSAVAT, F. 2000. Force-based motion editing for locomotion tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 24-28*.

POPOVIC, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Proceedings of SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, A. Rockwood, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 11–20.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 1996*, ACM Press / ACM SIGGRAPH, H. Rushmeier, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 147–154.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, R. Cook, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 91–96.

VAN DE PANNE, M., AND FIUME, E. 1993. Sensor-actuator networks. In *Proceedings of SIGGRAPH 1993*, ACM Press / ACM SIGGRAPH, J. T. Kajiya, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 335–342.

WANG, T., DORDEVIC, G. S., AND SHADMEHR, R. 2001. Learning the dynamics of reaching movements results in the modification of arm impedance and long-latency perturbation responses. *Biological Cybernetics 85*, 6, 437–448.

WINTERS, J. M., AND CRAGO, P. E., Eds. 2000. Springer-Verlag.

WISE, S. P., AND SHADMEHR, R. 2002. *Motor Control*. Elsevier Science.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Proceedings of SIGGRAPH 1988*, ACM Press / ACM SIGGRAPH, J. Dill, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 159–168.

WITKIN, A., AND POPOVIC, Z. 1995. Motion warping. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, R. Cook, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 105–108.

WOLPERT, D. M., AND GHAHRAMANI, Z. 2000. Computational principles of movement neuroscience. *Nature Neuroscience 3*, 1212–1217.

ZORDAN, V. B., AND HODGINS, J. K. 1999. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Graphics Forum (Proceedings of Eurographics 1999)*, 13–22.