

# Motion Perturbation Based on Simple Neuromotor Control Models

KangKang Yin\*, Michael B. Cline\*, Dinesh K. Pai\*†

\*University of British Columbia, Vancouver, Canada

†Rutgers, the State University of New Jersey, Piscataway, NJ

{kkyin|cline|pai}@cs.ubc.ca

## Abstract

*Motion capture is widely used for character animation. One of the major challenges of this technique is how to modify the captured 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 movement. 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 animation where the character reacts to unexpected external forces in a natural way (e.g., when the character is hit by a flying object), and still retain the quality of the captured motions. This technique is useful for applications such as interactive sports video games.*

## 1 Introduction

Motion capture is widely used today for realistic and stylistic human animation. However, the amount of motion we can capture does not meet our needs. In applications such as sports video games, the lack of variation between similar motions, or the lack of changes due to novel situations greatly reduces the sense of reality.

Dynamic simulation can generate responsive online motions. However, human simulation still lacks realism. We believe this is primarily due to the lack of human motion control models, and not to inadequacies in physical modelling. After all, an industrial robot obeys the same laws of physics as a human, and also moves in a physically plausible way. 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

creating realistic motor control is to manually design motor controllers for various motor skills. This turns out to be even harder than keyframing animation. The problem would be reduced if at least part of the control could be estimated from motion capture data.

This situation suggests we should 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, which encapsulates much knowledge of how humans control their movements, as well as rich style information. The other possibility is to borrow research in human motor control from neuroscience, biomechanics and other related movement sciences. Human movement control is still an active and contentious research area of its own, the mystery of which 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.

We propose a method of incorporating a simple human neuromuscular control model into dynamic simulation systems. Original motion capture animations can be modified adaptively, according to small unexpected disturbances arising from a dynamically changing environment.

## 2 Related Work

Highly skilled specialists have successfully designed motor controllers for dynamic human simulation by hand [12]. Such controllers are composable using machine learning techniques [7]. Incorporating motion capture into dynamic simulations makes the control problem easier to solve, which, in its simplest form, directly uses a tracking controller connected to a motion capture device [25].

Spacetime constraints (SC) [24, 8, 19] put the motion editing problem into a constrained optimization framework. SC combines kinematic keyframing (space constraints) with dynamic simulation (time constraints). Conceptually, a constrained optimizer is used as the motor controller. Motor learning techniques [20, 10] for physically based animation lead to the discovery of 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 for more complex behaviors.

To our knowledge, the work most similar to ours in Computer Graphics is [26]. We will differentiate our work from theirs after we discuss the characteristics of biological motor control systems in the next section.

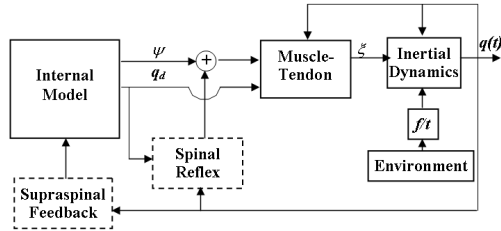
### 3 A Biologically based Motor Control Model

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 30-50ms. These are very large delays when compared with the total movement duration of very fast (150ms) to intermediate (500ms) movements [15]. 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, motor control can be executed using feedforward muscle forces, in an almost open-loop manner [15, 17]. This explains why our movements show highly stereotyped and stylized patterns, although almost any task can, in principle, be achieved in infinitely different ways. This also explains the observation that well-trained movements exhibit relatively low joint stiffness, while during motor learning the stiffness is higher. Therefore, speed and accuracy are lower due to lack of good internal models [9].

The intrinsic mechanical properties of muscles and tendons produce proportional (stiffness) and derivative (viscosity) feedback forces without delay [11]. Our model uses this muscle property as a low-gain feedback controller to stabilize the limb along the desired trajectory. Muscle force-length relationships can be quite complex, but it is well known that muscle stiffness increases with generated force [23]. A simple model of this non-linear relationship is the so-called bilinear model of muscle impedance [13, 22]. This model implies that effective stiffness is proportional to neural input, and hence, to generated muscle forces. We assume that muscle viscoelasticities also increase in a similar way and are small for well-trained movements.

Figure 1 shows our reference motor control model [14, 21]. The internal model is treated as a black box whose output is the feedforward motor command  $\psi$  and desired



**Figure 1. Motor control model for motion transformation upon unexpected disturbances**

trajectory  $q_d$ . The muscle-tendon system is driven by the motor command and generates the force  $\xi$ . 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 supraspinal feedback involving the brain stem and the brain which has even longer 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 re-plan the motion. The trajectory is restored by 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 motions, muscle feedback has low gain. So we can estimate the muscle force  $\xi$  by inverse dynamics from motion capture data, and use  $\xi$  as an approximation of feedforward command  $\psi$  (section 5.1).

We now contrast our approach to that of [26], which simulates motion capture-driven motions that hit and react. They use a high-gain tracking controller when there is no external impact. High stiffness parameters make the simulations appear overly strong and inflexible when contact is made. So the gains of the affected joints have to be reduced explicitly to allow the dynamics of the impact to influence the motion in a natural manner. Their high-gain tracking controller is similar to what is used in robotics. Based on all the previous discussions, we know that biological systems use a feedforward controller to do most of the work, and only use low-gain feedback to deal with neuromuscular noise. Upon perturbation, the muscle stiffness actually increases instead of decreasing, due to the stretch reflex (the most important and most studied spinal reflex [14]). Based on the fundamental characteristics of biological systems, we use feedforward forces plus low-gain feedback, which need only correct simulation drifts. Upon perturbation, simple muscle stiffness models can already cope with reactions and restorations fairly well, without a manually designed gain-scheduling controller, as used in [26]. Finally, our system simulates perturbations in real-time, while [26] is an off-line

system (at time of publication).

In general, impacts and other disturbances to the upper body may require lower body motion to change as well, to maintain balance and posture. [26] demonstrates how the lower body can be controlled separately by a balance controller that tries to keep the center of mass within the support polygon at all times. Our motions are very dynamic and athletic (i.e., football), and involve dramatic foot movements, so we cannot adopt the above approach directly. For such motions, maintaining balance in a “human-like” way, close to the quality of the original motion capture data, remains an unsolved problem.

## 4 Dynamic Simulation of Mocap Data

We develop a general-purpose rigid body simulation system, and extend it to meet the requirements of motor control [4, 5]. The simulator is fast enough to simulate the dynamics of our 54 DOF (degrees of freedom) character in real time. Our simulator is based on a Lagrange multiplier approach for computing constraint forces, inspired by the work of Baraff [3]. We extend this approach by allowing the simulator to solve both forward and inverse dynamics problems. We also introduce some new matrix notations that we use to describe the muscle forces, and include them in our equations of motion. Contact is managed as a linear complementarity problem. Since it is not directly related to motor control, we refer the reader to [4] for details.

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

$$\begin{bmatrix} \mathbf{M} & -\mathbf{J}^T & -\mathbf{H}^T \\ \mathbf{J} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \lambda \\ \tau \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{ext} \\ 0 \end{bmatrix} \quad (1)$$

Here  $\mathbf{M}$  is the mass-inertia matrix of the bodies in the system (a block diagonal matrix with each block corresponding to one body),  $\mathbf{J}$  is the constraint Jacobian,  $\mathbf{H}$  is a matrix whose rows span the space of possible muscle forces,  $\mathbf{a}$  is the acceleration vector of the bodies,  $\lambda$  is a vector of Lagrange multipliers,  $\tau$  is the corresponding set of muscle force multipliers, and vector  $\mathbf{f}_{ext}$  contains external forces such as gravity and coriolis forces.

Equation 1 is a unified expression that is true for both forward and inverse dynamics. We can rearrange this equation to reflect the known and unknown values in these two types of problems. Details can be found in [4, 6].

### 4.1 Forward Dynamics

In forward dynamics, the muscle force multipliers  $\tau$  are known quantities. Moving  $\mathbf{H}^T \tau$  to the right hand side of

equation 1, and then discretizing gives

$$\begin{bmatrix} \mathbf{M} & -\mathbf{J}^T \\ \mathbf{J} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{t+h} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{v}_t + h\mathbf{k} \\ 0 \end{bmatrix} \quad (2)$$

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  $\mathbf{k} = \mathbf{f}_{ext} + \mathbf{H}^T \tau$ . 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 [2, 5], where after each simulation step we make a small correction to the position of the bodies so that the constraints are maintained.

In order for the techniques in this paper to work for large time steps and achieve real-time simulation, an implicit integrator is essential. We use an implementation of the linearly implicit time stepping method [1]. Fortunately this requires only small modifications to an explicit integrator. 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. We calculate them using automatic differentiation techniques [18].

### 4.2 Inverse Dynamics

Inverse dynamics is the process of finding a set of forces that explain a given motion. The inverse dynamics equations we solve are another form of equation 1. We move  $\mathbf{M}\mathbf{a}$  to the right hand side of the equation (because the acceleration is a known quantity). Assuming the constraint equations  $\mathbf{J}\mathbf{v} = \mathbf{0}$  are satisfied by the given motion, we no longer need the second row of equation 1. We are left with

$$\begin{bmatrix} \mathbf{J}^T & \mathbf{H}^T \end{bmatrix} \begin{bmatrix} \lambda \\ \tau \end{bmatrix} = \mathbf{M}\mathbf{a} - \mathbf{f}_{ext} \quad (3)$$

If we can estimate the mass properties of the bodies of our articulated figure, along with the accelerations of its component bodies, then equation 3 can be solved to determine the muscle forces multipliers  $\tau$ .

## 5 Integrating Motor Control with Dynamic Simulation

The integration of our motor control method with dynamic simulation 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  $\tau$  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  $\tau$  using equation 3.

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

One problem in evaluating equation 3 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).

A substantial problem is that we do not know the ground reaction forces for the full body motions we capture. For this reason, and because maintaining balance during highly athletic motions remains an unsolved problem, we simply constrain the root joint to move along the motion capture path. We can thus omit the contact dynamics with the floor.

Using equation 3, we precompute feedforward torque multipliers for each of the frames in the animation before beginning the forward simulation. The total feedforward torque is given by  $H^T \tau$ . The dimension of  $\tau$  is the sum of the degrees of freedom, which we denote with  $n$ . For the next section, we need to break this down into smaller components  $\tau_1, \dots, \tau_n$ , each of which corresponds to one DOF of the joints.

$$H^T \tau = \mathbf{h}_1 \tau_1 + \mathbf{h}_2 \tau_2 + \dots + \mathbf{h}_n \tau_n \quad (4)$$

We store  $\tau_i$ ’s instead of  $\mathbf{h}_i \tau_i$ ’s, because the orientation of the joints with respect to the world, that is,  $\mathbf{h}_i$ ’s, may be different in forward simulation than in the original motion capture animation.

## 5.2 Combining Feedback and Feedforward Torques During Forward Simulation

Individual feedback torques are calculated for each DOF of all of the joints. Let  $\theta_1, \theta_2, \dots, \theta_n$  be joint angles corresponding to each DOF, and  $\dot{\theta}_1, \dot{\theta}_2, \dots, \dot{\theta}_n$  be joint velocities. The joint angles  $\theta_{d1}, \theta_{d2}, \dots, \theta_{dn}$  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  $\gamma_i = \mathbf{h}_i |\tau_i| \left( k_{s_i} (\theta_{di} - \theta_i) + k_{d_i} (\dot{\theta}_{di} - \dot{\theta}_i) \right)$ . where  $k_{s_i}$  and  $k_{d_i}$  are the stiffness and damping constants for each DOF. The measurement of these parameters has been conducted in biomechanics for quite some time [16], yet a lookup table for humans is still unavailable. Luckily, we do not need high accuracy for animation purposes. Basically, we experimentally determine only one pair of these constants, and scale

other gains according to the moment of inertia of the chain of bodies affected by that joint [26]. Note that the stiffness is proportional to the magnitude of the muscle torque multiplier,  $|\tau_i|$ , as observed empirically in muscle biomechanics.

The feedforward torques are computed by  $\psi_i = \mathbf{h}_i \tau_i$ . The total muscle torques are given by the sum of the feedforward and feedback torques:  $\xi = \sum_{i=1}^n (\gamma_i + \psi_i)$ . The muscle torques are added into the dynamics equation along with any other external forces, such as gravity.

## 6 Results

The skeleton model we use in our simulation and rendering has 54 DOF in total. The kinematic root is a joint located roughly at the Lumbosacral angle of the spine. The dynamic simulation runs in real-time on a dual-CPU (1.78 GHz Intel) machine when there is no contact, or when there are simple contacts. The frame rate drops a little bit when complex contacts happen. We also employed hardware rendering techniques to off-load the graphics rendering from CPU to GPU.

We perform perturbation experiments on captured arm motions as well as full body motions from football games. In both cases, the simulated skeleton responds to external disturbances and restores itself to the original motion naturally. Figure 2 shows motion perturbation on a sequence of full body motion.

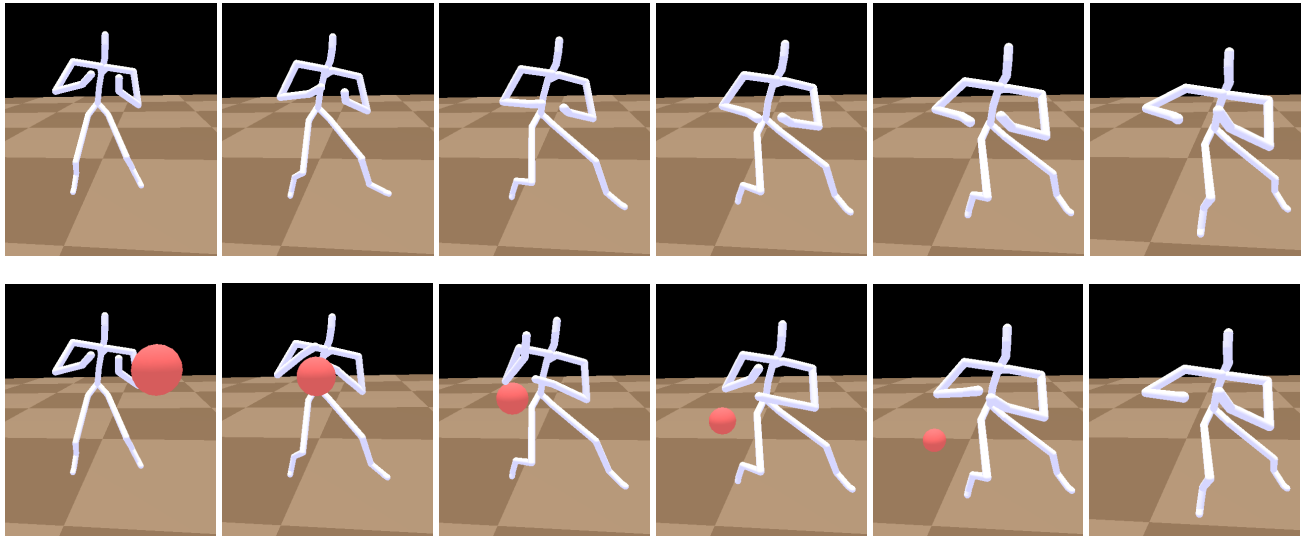
## 7 Conclusions and Future Work

Motor control is one of the major challenges in physically based human simulation. To our knowledge, this paper is the first work that tries to address this problem by explicitly incorporating human neuromotor control models into a human simulation system. We test our approach with motion perturbation tasks on motion capture data, and the results are promising.

The main limitation of this work is that the implemented motor control model is currently very simple: spinal and supraspinal feedback pathways are missing. This limits us to only addressing disturbances that do not endanger balance, and are recoverable by muscle-tendon feedback. We plan to address the longer latency (and also more complex and less well understood) feedback pathways in future work.

Despite the simplicity of the implemented motor control model, animation with realistic responses to small perturbations can be generated, in real time. Applications such as interactive video games can thus benefit from an enriched motion repertoire.

**Acknowledgements:** The authors would like to thank Electronic Arts for providing us with motion capture data.



**Figure 2. First row: an original motion. Second row: its perturbed motion.**

## References

- [1] M. Anitescu and F. Potra. A time-stepping method for stiff multibody dynamics with contact and friction. *International Journal for Numerical Methods in Engineering*, 55:753–784, 2002.
- [2] U. Ascher, H. Chin, L. Petzold, and S. Reich. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Mechanics of Structures and Machines*, 23:135–158, 1995.
- [3] D. Baraff. Linear-time dynamics using lagrange multipliers. In H. Rushmeier, editor, *Proceedings of SIGGRAPH 1996*, pages 137–146, 1996.
- [4] M. Cline. Rigid body simulation with contact and constraints. Master’s thesis, University of British Columbia, 2002.
- [5] M. Cline and D. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.
- [6] M. Cline, K. Yin, and D. Pai. Motion perturbation based on simple neuromotor control models. Technical Report TR-2002-03.
- [7] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, pages 251–260, 2001.
- [8] M. Gleicher. Retargetting motion to new characters. In *Proceedings of SIGGRAPH 1998*, pages 33–42, 1998.
- [9] H. Gomi and M. Kawato. Human arm stiffness and equilibrium-point trajectory during multi-joint movement. *Biological Cybernetics*, 76:163–171, 1997.
- [10] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 1998*, pages 9–20, 1998.
- [11] S. Hall. *Basic Biomechanics*. Mosby, second edition, 1998.
- [12] J. Hodgins, W. Wooten, D. Brogan, and J. O’Brien. Animating human athletics. In *Proceedings of SIGGRAPH 1995*, pages 71–78, 1995.
- [13] N. Hogan. Mechanical impedance of single and multi-articular systems. In J. Winters and S. Woo, editors, *Multiple Muscle Systems: Biomechanics and Movement Organization*, chapter 9, pages 149–64. Springer-Verlag, 1990.
- [14] E. Kandel, J. Schwartz, and T. Jessell. *Principles Of Neural Science*. McGraw–Hill, fourth edition, 2000.
- [15] M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9:718–727, 1999.
- [16] M. Latash and V. Zatsiorsky. Joint stiffness: Myth or reality. *Human Movement Science*, 12:653–692, 1993.
- [17] F. Mussa-Ivaldi. Modular features of motor control and learning. *Current Opinion in Neurobiology*, 9:713–717, 1999.
- [18] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.
- [19] Z. Popovic and A. Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 1999*, pages 11–20, 1999.
- [20] M. van de Panne and E. Fiume. Sensor-actuator networks. In *Proceedings of SIGGRAPH 1993*, pages 335–342, 1993.
- [21] T. Wang, G. Dordevic, and R. Shadmehr. Learning the dynamics of reaching movements results in the modification of arm impedance and long-latency perturbation responses. *Biological Cybernetics*, 85(6):437–448, 2001.
- [22] J. Winters and P. Crago, editors. *Biomechanics and Neural Control of Posture and Movement*. Springer-Verlag, 2000.
- [23] S. Wise and R. Shadmehr. *Motor Control*. Elsevier Science, 2002.
- [24] A. Witkin and M. Kass. Spacetime constraints. In *Proceedings of SIGGRAPH 1988*, pages 159–168, 1988.
- [25] V. Zordan and J. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Graphics Forum (Proceedings of Eurographics 1999)*, pages 13–22, 1999.
- [26] V. Zordan and J. Hodgins. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 89–96, July 2002.