Visual Comput (2005)
DOI 10.1007/s00371-005-0284-4

ORIGINAL ARTICLE

Vincent De Sapio
James Warren
Oussama Khatib
Scott Delp

Simulating the task-level control of human motion: a methodology and framework for implementation

Published online: 24 May 2005

Vincent De Sapio · James Warren · Oussama Khatib
Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, CA 94305, USA
e-mail: {vdesap, warren, khatib}@robotics.stanford.edu
Send offprint requests to: V. De Sapio, Clark Center Room E100, 318 Campus Dr., Stanford, CA 94305-5450

Scott Delp
Neuromuscular Biomechanics Laboratory, Mechanical Engineering and Bioengineering Departments
Stanford University, Stanford, CA 94305, USA
e-mail: delp.stanford.edu

Abstract A task-level control framework is proposed for providing feedback control in the simulation of goal-directed human motion. An operational space approach, adapted from the field of robotics, is used for this purpose. This approach is augmented by a significant new extension directed at addressing the control of muscle-driven systems. Task/posture decomposition is intrinsically exploited, allowing human musculoskeletal properties to direct postural behavior during the performance of a task. This paper also describes a simulation architecture for generating musculoskeletal simulations of human characters. The evolving capabilities of the collective environment are directed toward autonomously generating realistic motion control for virtual actors in interactive computer graphics applications, as well as synthesizing the control of human-like motion in robotic systems.

Keywords Human animation · Musculoskeletal dynamics · Robotics · Task-level control

1 Introduction

In recent years, there has been an emerging desire to synthetically generate human-like motion in both simulated and physical settings. In the computer graphics community, this desire is directed toward autonomously generating realistic motion for virtual actors [5, 6, 29]. The intent is to direct these virtual actors using high-level goal-directed commands for which low-level motion control is automatically generated.

Motivated by similar desires, the robotics community seeks a high-level control framework for robotic systems. With the recent advent of complex humanoid robots [19, 25] this challenge grows more demanding. Consistent with their anthropomorphic design, humanoid robots are intended to operate in a human-like manner within man-made environments and to promote interaction with their biological counterparts. Common control strategies involve generating joint space trajectories [19] or learning specific motions [10], but these approaches require extensive motion planning computations and do not generalize well to related tasks.

The challenge of synthesizing low-level human motion control from high-level commands can be addressed by integrating approaches from the biomechanics and robotics communities. The biomechanics community has investigated the phenomenon of neuromuscular dynamics and control through the use of computational muscle models [3, 4, 31]. This characterization allows us to describe muscle strength limitations, activation delays, and overall muscle contraction dynamics. Properly accounting

for these characteristics is critical to authentically simulating human motion since all human motion is rooted in, and bounded by, physiological capabilities. In a complementary manner, the robotics community has investigated the task-level feedback control of robots using the operational space approach [12, 13, 15]. This approach recasts the dynamics of the robotic system into a relevant *task space* description. This provides a natural mechanism for specifying high-level motion commands that can be executed in real-time using feedback control.

While the use of computational musculoskeletal models and the task-level control of robots have been researched in the past, the integration of these approaches into a unified framework offers a promising methodology for synthesizing goal-directed human-like motion control. Recent work [15] addresses multi-objective task-level control of humanoid systems. However, the authors do no not address the biomechanical factors that influence human motion control. Such factors, like musculoskeletal dynamics, form an important part of emulating human motion in a synthetic environment.

This paper focuses on the biomechanical aspects of human motion in order to provide a physiological framework for simulating human motion. We present a control architecture that integrates elements of biomechanical and robotic approaches. This control architecture extends the operational space approach by addressing the control of muscle-driven systems. This extension complements other work done in operational space control of humanoid systems [15] that addresses multiple task objectives but not the physiological basis for motion.

In addition to the new control framework, this paper presents a simulation architecture for the task-level control of human-like motion. This consists of a simulation implementation that generates feed-forward musculoskeletal simulations of human-like characters. This is built upon a core environment that has already been developed for the physics-based simulation of robotic systems. The evolving capabilities of the overall environment are facilitated by an object-oriented design, which provides a natural mechanism for connecting the musculoskeletal plant to task-level controllers. The environment offers real-time simulation functionality, allowing the user to control and interact with many degree-of-freedom muscle-driven models in a task-level setting.

2 Previous work

The operational space framework [12, 13] provides the point of departure for the task-level control approach that will be addressed in this paper. This framework provides a natural vocabulary for studying human motion control. It does so by providing dynamic models at the task level and structures for decoupled task and posture control. This allows for posture objectives to be controlled without dynamically interfering with the primary task(s). More recently a multi-objective formulation has been proposed for the control of humanoid systems [15]. The efficacy of the operational space framework has been demonstrated for robotic systems; however, to date, it has only had limited application for biomechanical systems [27]. Specifically, past work in formulating operational space control, including the multi-objective formulation [15], makes no accommodation for muscle-driven control. This paper extends the application of the operational space approach to biomechanical simulations.

Several tools have emerged in the biomechanics community for studying the dynamics and neuromuscular control of movement. Many biomechanics labs have made use of SIMM (Software for Integrated Musculoskeletal Modeling) [4], a modeling environment for defining skeletal topology, muscle geometry, and joint kinematics. When used in conjunction with SD-FAST (Symbolic Dynamics) [9] to generate the multi-body equations of motion for the skeletal system, the user can simulate the feed-forward dynamic response of the musculoskeletal system to neural inputs. There are no native control capabilities in SIMM, so users must specify the control via open-loop using their own feed-forward optimization routine [22] or via closed-loop using their own feedback control routine [28]. In addition to SIMM, there have been several other musculoskeletal simulation systems described in the literature [8, 17, 18, 23].

3 Musculoskeletal model

The musculoskeletal model presented here involves a description of the skeletal system as a rigid multi-body system spanned by a set of musculotendon actuators. The musculotendon actuator model consists of a standard two-state Hill-type model [26, 30]. A stiff tendon simplification of the two-state model will also be presented.

While this musculoskeletal model has been broadly used, Sect. 5 describes a unique integration of this model into a task-level control framework. This involves encoding musculoskeletal parameters into a postural field to facilitate control.

3.1 Skeletal dynamics

The skeletal system is modeled as a system of constrained rigid bodies whose configuration is described by a set of generalized coordinates, $\boldsymbol{q}$. These generalized coordinates are usually taken to be the joint angles between limb segments but may also involve other displacement parameters. The n equations of motion can thus be represented in standard form as

$$\boldsymbol{\Gamma} = \boldsymbol{A}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}), \tag{1}$$

where $\boldsymbol{\Gamma}$ is the set of joint torques, $\boldsymbol{A}(\boldsymbol{q})$ is the system mass matrix, $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the vector of centrifugal and Coriolis terms, and $\boldsymbol{g}(\boldsymbol{q})$ is the vector of gravity terms. For conciseness, we will often refrain from explicitly denoting the functional dependence of these quantities on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. This practice will also be employed with other quantities throughout the paper.

3.2 Muscle kinematics

Having defined a dynamical model of the skeletal system, we may now assign a system of musculotendon actuators to it. For the purposes of this model, we will assume that all musculotendon lengths, $\boldsymbol{l}$, can be uniquely determined from the system configuration, $\boldsymbol{q}$. That is, $\boldsymbol{l} = \boldsymbol{l}(\boldsymbol{q})$. As a consequence of this assumption, differential variations in $\boldsymbol{l}$ are given by

$$\delta \boldsymbol{l} = \boldsymbol{L}(\boldsymbol{q})\, \delta \boldsymbol{q}, \tag{2}$$

where $\boldsymbol{L}(\boldsymbol{q})$ is the muscle Jacobian. From the principle of virtual work, we conclude that

$$\boldsymbol{\Gamma} = -\boldsymbol{L}^T \boldsymbol{f}_T, \tag{3}$$

where $\boldsymbol{f}_T$ is the vector of net muscle forces (active and passive components). The negative sign is due to the convention of taking contractile muscle forces as positive.

3.3 Neuromuscular dynamics

Neuromuscular dynamics refers to the time-evolution of muscle states (length and force) in response to neural excitation (control input) [11, 21]. This net dynamic effect can be divided into activation dynamics and musculotendon contraction dynamics. A Hill-type active state model [26, 30] has been used to model these effects.

3.3.1 Activation dynamics

Activation dynamics refers to the process of muscle activation in response to neural excitation. This process can be modeled by the following equation of state written in terms of the activation, a:

$$\dot{a} = \frac{u - a}{\tau(u, a)}, \tag{4}$$

where $u \in [0, 1]$ is the neural input. The term $\tau(u, a)$ is a time constant given by

$$\tau(u, a) = \begin{cases} (\tau_a - \tau_d)u + \tau_d & \text{for } u \geqslant a \\ \tau_d & \text{for } u < a \end{cases}, \tag{5}$$

where τ_a and τ_d are the activation and deactivation time constants, respectively.

3.3.2 Contraction dynamics

Musculotendon contraction dynamics refers to the process of force generation in the muscle based on muscle contraction, rate of contraction, and activation. A number of different active state models exist. One such model is formulated with muscle length, l_M, as the state [26, 30]. Figure 1 depicts the configuration of forces in this model. In addition to an active element, there is a passive viscoelastic element (in parallel) and an elastic tendon element (in series). The relative angle associated with the muscle fibers, α, is referred to as the pennation angle.

Based on the geometry of Fig. 1, we have the following relationships:

$$\begin{aligned} l(\boldsymbol{q}) &= l_M \cos\alpha + l_T \\ \dot{l}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \dot{l}_M \cos\alpha + \dot{l}_T \end{aligned} \tag{6}$$

and the following force equilibrium equation:

$$f_T = (f_A + f_P)\cos\alpha. \tag{7}$$

Using this force equilibrium equation in conjunction with constitutive relationships (Fig. 2) we can express an equation of state in the following functional form:

$$\dot{l}_M = \dot{l}_M(l(\boldsymbol{q}), \dot{l}(\boldsymbol{q}, \dot{\boldsymbol{q}}), l_M, a). \tag{8}$$

For a system of r musculotendon actuators, we can express the following system of $2r$ first-order state equa-

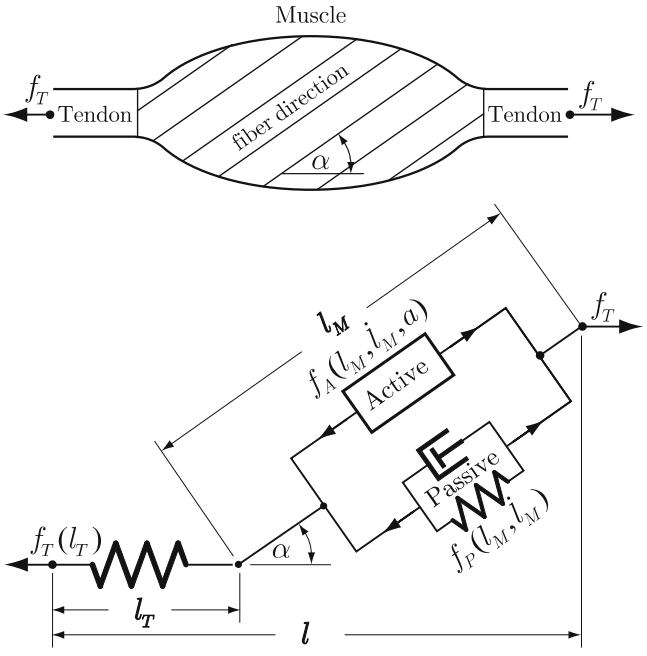


Fig. 1. Active state musculotendon model. The active contractile element and passive viscoelastic element are in parallel. The passive elastic tendon element is in series

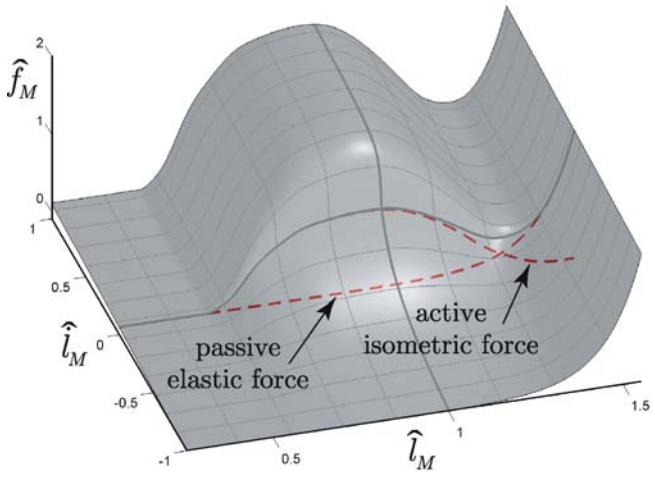


Fig. 2. Force-length-velocity relationship showing muscle force, $\widehat{f}_M$, as a function of muscle length, $\widehat{l}_M$, and rate of contraction, $\widehat{\dot{l}}_M$ (all quantities are normalized)

tions:

$$\dot{\boldsymbol{a}} = \begin{pmatrix} 1/\tau_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & 1/\tau_r \end{pmatrix} \begin{pmatrix} u_1 - a_1 \\ \vdots \\ u_r - a_r \end{pmatrix}, \tag{9}$$

and

$$\dot{\boldsymbol{l}}_M = \dot{\boldsymbol{l}}_M(\boldsymbol{l}(\boldsymbol{q}), \dot{\boldsymbol{l}}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \boldsymbol{l}_M, \boldsymbol{a}) \tag{10}$$

3.4 Musculoskeletal dynamics

Combining neuromuscular dynamics with multi-body skeletal dynamics allows us to model the overall musculoskeletal system. Given a set of neural inputs, $\boldsymbol{u}$, a set of muscle activations, $\boldsymbol{a}$, arise as dictated by the activation dynamics. Based on these activations, as well as the skeletal configuration, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, a set of muscle forces, $\boldsymbol{f}_T$, arises. These muscle forces are related to the joint torques, $\boldsymbol{\Gamma}$, through the muscle Jacobian, $\boldsymbol{L}$. The skeletal dynamics are driven by these joint torques, resulting in motion of the system. This overall process is described by the feed-forward path of Fig. 3. This path, however, does not address the feedback mechanisms used by the central nervous system to arrive at appropriate compensation for controlling the musculoskeletal plant.

3.4.1 Stiff tendon model

If we make the assumption that the tendon is infinitely stiff, l_M is no longer an independent state, but rather, it is algebraically related to the overall musculotendon length, $l(\boldsymbol{q})$. This provides a useful simplification to the full musculoskeletal dynamics described in the previous sections. If we define the muscle saturation force, $f_S(\boldsymbol{q}, \dot{\boldsymbol{q}}) \triangleq f_A(\boldsymbol{q}, \dot{\boldsymbol{q}}, 1)$, we can then define the muscle force-activation gain, K_f, as the magnitude of force generation in the muscle per change in unit activation. Thus,

$$K_f \triangleq \frac{\partial f_T}{\partial a} = f_S \cos\alpha. \tag{11}$$

For a system of muscles, the muscle force-activation gain matrix is then

$$\boldsymbol{K}_f = \begin{pmatrix} f_{S_1}\cos\alpha_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & f_{S_r}\cos\alpha_r \end{pmatrix}, \tag{12}$$

and the muscle forces can now be expressed in terms of this gain:

$$\boldsymbol{f}_T(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{a}) = \boldsymbol{f}_P(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{K}_f(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \boldsymbol{a}, \tag{13}$$

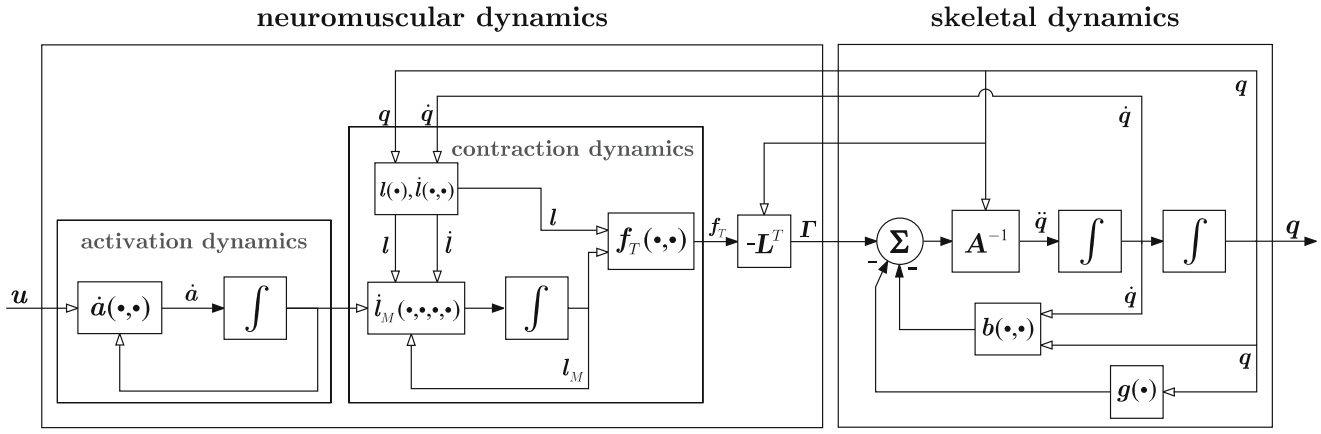


Fig. 3. Musculoskeletal system (feed-forward path). Neural excitations provide input to the activation dynamics. Output of the activation dynamics provides input to the contraction dynamics. Output of the contraction dynamics provides input to the skeletal dynamics through the joint torques

where

$$\boldsymbol{f}_P = \left(f_{P_1}\cos\alpha_1 \cdots f_{P_r}\cos\alpha_r\right)^T. \tag{14}$$

As with the musculotendon forces, it is useful to express our generalized forces using a gain relationship. To this end, we will define the muscle torque-activation gain matrix,

$$\boldsymbol{K}_\Gamma \triangleq -\boldsymbol{L}^T\boldsymbol{K}_f. \tag{15}$$

The generalized forces can now be expressed in terms of this gain,

$$\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{a}) = \boldsymbol{\Gamma}_P(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{K}_\Gamma(\boldsymbol{q},\dot{\boldsymbol{q}})\,\boldsymbol{a}, \tag{16}$$

where

$$\boldsymbol{\Gamma}_P = -\boldsymbol{L}^T\boldsymbol{f}_P. \tag{17}$$

The overall musculoskeletal dynamics can then be expressed as

$$\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{a}) = \boldsymbol{A}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{b}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}). \tag{18}$$

4 Task-level control framework

Within the context of our control framework, a task is any formal description of desired activity that can explicitly be represented as a function of the generalized coordinates. This could be as simple as specifying the position of a limb to be in a certain location. Multiple tasks can be combined into a single task definition, as long as they are kinematically consistent with each other. The *task coordinates* will be denoted by $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{q})$. The Jacobian, $\boldsymbol{J}(\boldsymbol{q})$, of the task is defined as

$$J_{ij} \triangleq \frac{\partial x_i}{\partial q_j} \tag{19}$$

Defining the dynamically consistent inverse of the Jacobian as [12] [13]

$$\bar{\boldsymbol{J}} \triangleq \boldsymbol{A}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{A}^{-1}\boldsymbol{J}^T)^{-1}, \tag{20}$$

we can map the configuration space description of the skeletal dynamics (Eq. 1) into an operational space description:

$$\begin{array}{c} \bar{\boldsymbol{J}}^T\left(\boldsymbol{A}\ddot{\boldsymbol{q}} + \boldsymbol{b} + \boldsymbol{g} = \boldsymbol{\Gamma}\right) \\ \Downarrow \\ \boldsymbol{\Lambda}(\boldsymbol{q})\,\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{p}(\boldsymbol{q}) = \boldsymbol{f} \end{array}. \tag{21}$$

This process provides us a description of the dynamics in *task coordinates* rather than generalized coordinates (while generalized coordinates are still present in Eq. 21, the inertial term involves task space accelerations rather than generalized accelerations). This is useful for control applications, since we can frame our control problem in terms of the relevant task coordinates, $\boldsymbol{x}$, that we wish to control using a relevant operational space force, $\boldsymbol{f}$. The terms in Eq. 21 are defined as the operational space mass matrix,

$$\boldsymbol{\Lambda}(\boldsymbol{q}) \triangleq \left(\boldsymbol{J}\boldsymbol{A}^{-1}\boldsymbol{J}^T\right)^{-1}, \tag{22}$$

the operational space centrifugal-Coriolis vector,

$$\boldsymbol{\mu}(\boldsymbol{q},\dot{\boldsymbol{q}}) \triangleq \bar{\boldsymbol{J}}^T\boldsymbol{b} - \boldsymbol{\Lambda}\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \tag{23}$$

and the operational space gravity vector,

$$\boldsymbol{p}(\boldsymbol{q}) \triangleq \bar{\boldsymbol{J}}^T\boldsymbol{g}. \tag{24}$$

We will always take the dimension of the task to be less than or equal to the dimension of the configuration space. In the specific case that the dimension of the task is strictly less than the dimension of the configuration space, we refer to the system as redundant with respect to the task (see Fig. 4). In this case, the operational space dynamics do not fully describe the system behavior, and $\boldsymbol{f}$ does not fully describe the system control. However, the null space of the task is associated with a complementary posture space of motion that does not interfere with the task [13]. This posture space may be utilized to satisfy other criteria through the following decomposition of the control torques:

$$\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_{task} + \boldsymbol{\Gamma}_{posture} = \boldsymbol{J}^T\boldsymbol{f} + \left(\boldsymbol{I} - \boldsymbol{J}^T\bar{\boldsymbol{J}}^T\right)\boldsymbol{\Gamma}_o. \tag{25}$$

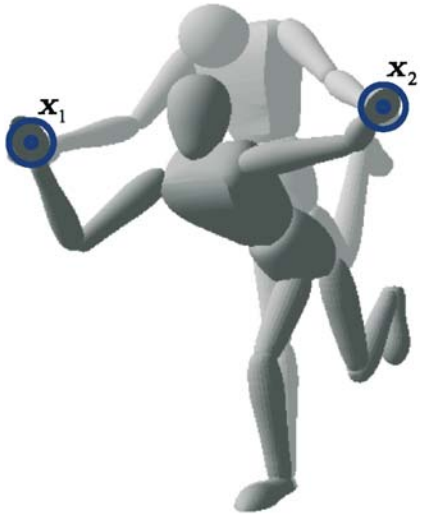


Fig. 4. A task description illustrating two corresponding task-consistent postures. Redundancy with respect to task introduces task dynamics as well as posture dynamics (simulation sequence created by L. Sentis)

A more concise expression of this is

$$\boldsymbol{\Gamma}_{task} + \boldsymbol{\Gamma}_{posture} = \boldsymbol{J}^T \boldsymbol{f} + \boldsymbol{N}^T \boldsymbol{\Gamma}_o, \tag{26}$$

where

$$\boldsymbol{N}^T \triangleq \boldsymbol{I} - \boldsymbol{J}^T \bar{\boldsymbol{J}}^T. \tag{27}$$

Khatib et al. [15] considered a multi-objective formulation of operational space control, allowing for a hierarchy of separate tasks that may conflict with each other. This multi-objective approach will not be used for the exposition presented throughout the rest of this paper. The basic formulation summarized above will suffice, given an assumption of non-conflicting tasks.

5 Task-level musculoskeletal control

We now wish to apply a task/posture control decomposition to our musculoskeletal system. This entails choosing physiologically characteristic task and posture control structures that model certain aspects of the motion control behavior of the central nervous system. Because we are choosing feedback control structures, the system can be designed to behave in a robust manner in the presence of disturbances. This is important for real-time interaction with a physics-based simulation or a physical system. In the absence of such feedback mechanisms, feed-forward optimization approaches like dynamic optimization would be required. Such approaches have the drawback of being computationally costly, thereby precluding real-time execution. Additionally, due to the open-loop nature of the controls they generate, the behavior of the system is not robust in the presence of disturbances and external user interactions.

5.1 Task control

We will begin our task/posture control synthesis by choosing a task control structure. This entails designing a control response, $\boldsymbol{f}$. An artificial potential field, $U_t(\boldsymbol{x})$, described in task coordinates, serves as a straightforward control primitive. For example

$$U_t(\boldsymbol{x}) \triangleq \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_o)^T \boldsymbol{K}_x (\boldsymbol{x} - \boldsymbol{x}_o) \tag{28}$$

can be used as a simple quadratic potential field, relative to a goal location, $\boldsymbol{x}_o$. Seeking to minimize this potential, we generate the following control force:

$$\boldsymbol{f}^\star = -\nabla U_t = -\boldsymbol{K}_x (\boldsymbol{x} - \boldsymbol{x}_o). \tag{29}$$

Adding a damping term, we have

$$\boldsymbol{f}^\star = -\boldsymbol{K}_x (\boldsymbol{x} - \boldsymbol{x}_o) - \boldsymbol{K}_v \dot{\boldsymbol{x}} \tag{30}$$

as the control input of the decoupled system. The gain matrices, $\boldsymbol{K}_x$ and $\boldsymbol{K}_v$, can be chosen to reflect physiological properties, such as natural limb endpoint impedances. Using estimates (denoted by $\widehat{\cdot}$) of the operational space dynamic properties, we have the following dynamic compensation expression:

$$\boldsymbol{f} = \widehat{\boldsymbol{\Lambda}} \boldsymbol{f}^\star + \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{p}}. \tag{31}$$

This expression will be complemented by a posture expression, to be developed in the following section.

5.2 Posture control

We will complete our task/posture control synthesis by choosing a posture control structure. This entails designing a control input, $\boldsymbol{\Gamma}_p$, to serve as the null space term, $\boldsymbol{\Gamma}_o$, in Eq. 26. A potential field, $U_p(\boldsymbol{q})$, can be chosen to represent any of a number of different physiological measures. Specifically, where these measures relate to musculoskeletal properties, we will refer them as *muscular effort* measures. Our control then seeks to minimize U_p while satisfying the task.

The expression for $U_p(\boldsymbol{q})$ proposed the gravity terms [16] involves muscle strength capacities projected at the joint level and is given by

$$U_p(\boldsymbol{q}) = \sum_{i=1}^{r} w_i \left(\frac{g_i(\boldsymbol{q})}{\Gamma_{B_i}(\boldsymbol{q})} \right)^2, \tag{32}$$

where, depending on the sign of the gravity terms g_i, the term Γ_{B_i} is either the upper or lower muscle induced torque boundary about a joint i. These terms are computed using the stiff tendon musculoskeletal model described in Sect. 3.4.1. The w_i terms are weighting terms.

The physiological rationale for this measure was discussed in [16] and will only be briefly described here. In this study, a number of motion-capture experiments were performed on human subjects. The experimental postures associated with a given reaching task were compared with other postures in the kinematic null space of the task. The set of these postures comprise the self-motion manifold associated with a fixed task point, $\boldsymbol{x}(\boldsymbol{q}) = \boldsymbol{x}_o$. This set is given by $M(\boldsymbol{x}_o) \triangleq \{\boldsymbol{q} | \boldsymbol{x}(\boldsymbol{q}) = \boldsymbol{x}_o\}$. In [16], good correlation was observed between the experimental postures and the postures that minimized the effort measure of Eq. 32.

It is noted that by projecting muscle strength capacities to the joint level in a decoupled manner, as is done in Eq. 32, the cross-joint coupling associated with multi-articular muscles (muscles that span more than one joint) is ignored. Consequently, in this paper we propose an enhanced and more general muscle effort measure to that which was proposed in [16]. This measure is given by

$$U_p(\boldsymbol{q}) = \boldsymbol{g}(\boldsymbol{q})^T \left(\boldsymbol{K}_\Gamma \boldsymbol{K}_\Gamma^T \right)^{-1} \boldsymbol{g}(\boldsymbol{q}), \tag{33}$$

where $\boldsymbol{K}_\Gamma$ is the muscle torque-activation gain matrix defined in Eq. 15.

Using the measure of Eq. 33, experimental studies were performed and analyzed. Motion capture experiments were performed on five subjects who were instructed to move different weights to various target locations. Figure 5 depicts the results of a particular subject trial. The effort associated with the subject's nominal configuration, $\boldsymbol{q}_o$, at the target can be compared with the effort associated with other configurations in a subset of the self-motion manifold, $\boldsymbol{q} = [\boldsymbol{q}_1, \boldsymbol{q}_2]$, of the target.

Not all trials show as precise a correlation between the subject's nominal configuration and the minimum of the muscle effort curve, as can be seen in the trial of Fig. 5. However, for most trials, the minimum of the effort curve is typically within several degrees of the subject's nominal configuration. More experiments are planned; nevertheless, the results are promising in that they suggest good correlation between muscle effort minimization and subject motion. The reader may refer to the work of Khatib et al. [16] for a more extensive discussion of experimental validation with human subjects.

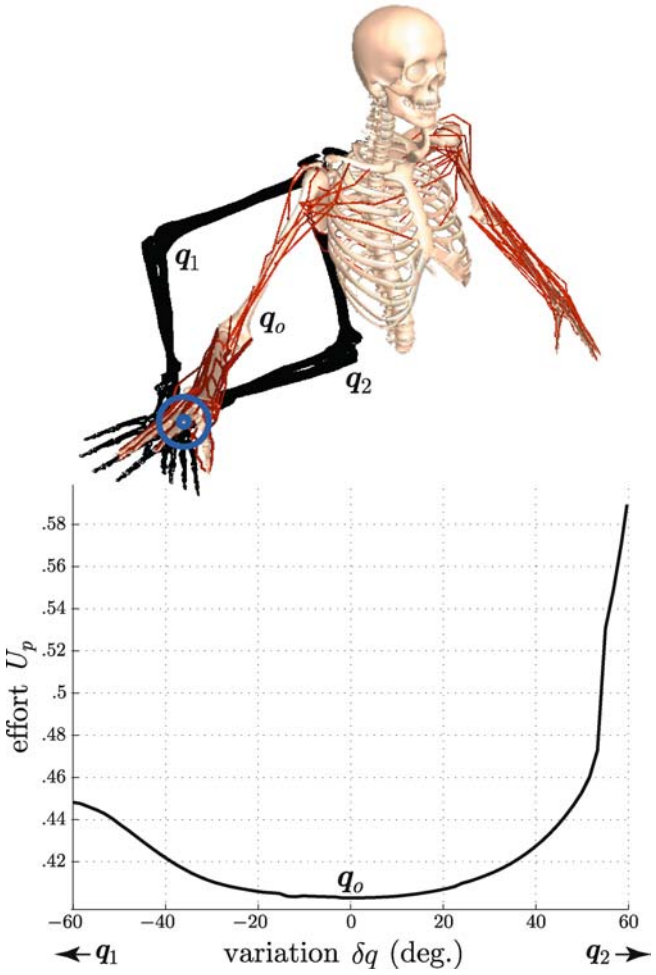


Fig. 5. Variation of muscle effort (Eq. 33) over the self-motion manifold for a given hand target location. The horizontal axis represents variation from the subject's nominal configuration, $\boldsymbol{q}_o$, in degrees along the self-motion manifold. For most trials, the minimum of the effort curve is typically within several degrees of the subject's nominal configuration

In the work presented by Khatib et al. [16], the approach employed with the muscle effort measure (Eq. 32) was effectively a kinematic one. While a prioritized task-level control framework is summarized, the authors do not explicitly integrate the musculoskeletal model into the dynamics and control setting. Further, the analysis done by Khatib et al. [16] is purely kinematic. By contrast, we wish to apply an effort-minimization approach to the control, where we seek to minimize the muscle effort potential in a task-consistent manner. To do this, we generate the following posture control torque:

$$\boldsymbol{\Gamma}_p = -\nabla U_p. \tag{34}$$

Adding a damping term and gains, we have

$$\boldsymbol{\Gamma}_p = -K_p \nabla U_p - K_d \dot{\boldsymbol{q}}. \tag{35}$$

5.3 Unified musculoskeletal task/posture control

With the individual task and posture control structures defined, we can implement the overall feedback control structure. In summary, our control torque is given by

$$\boldsymbol{\Gamma} = \boldsymbol{J}^T \boldsymbol{f} + \boldsymbol{N}^T \boldsymbol{\Gamma}_p, \tag{36}$$

where the operational space compensation is

$$\boldsymbol{f} = \widehat{\boldsymbol{\Lambda}} \boldsymbol{f}^\star + \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{p}}, \tag{37}$$

and the control input of the decoupled system is

$$\boldsymbol{f}^\star = -\boldsymbol{K}_x (\boldsymbol{x} - \boldsymbol{x}_o) - \boldsymbol{K}_v \dot{\boldsymbol{x}}. \tag{38}$$

Finally, the posture term is

$$\boldsymbol{\Gamma}_p = -K_p \nabla U_p - K_d \dot{\boldsymbol{q}}, \tag{39}$$

where U_p is the muscle effort measure given by Eq. 32, Eq. 33, or some other muscle-based potential. Figure 6 depicts a block diagram of the task-level neuromuscular controller described by Eqs. 36 through 39.

Figure 7 depicts responses associated with the task-level neuromuscular controller of Fig. 6. In this illustrative example, a muscle-driven redundant chain is controlled to move to a target. In one case, the distal muscles are stronger than the proximal muscles. In the other case, the proximal muscles are stronger than the distal muscles.

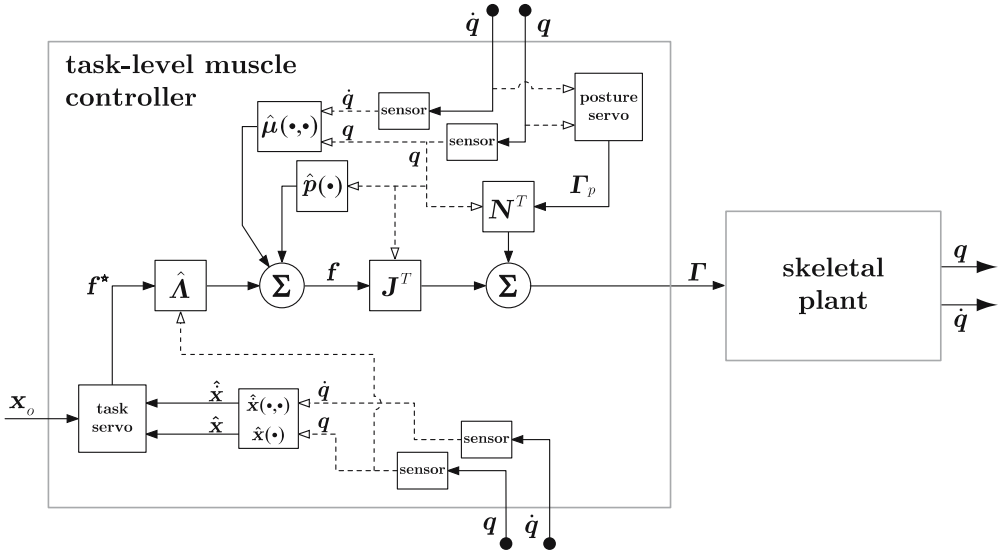


Fig. 6. Task-level feedback control employing a neuromuscular criterion for posture control. The *task servo* block represents a control law in task coordinates. This can be defined as the gradient of a task potential field plus a damping term (Eq. 38). Estimates of the operational space dynamic parameters, $\boldsymbol{\Lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{p}$ are used to provide the appropriate dynamic compensation, $\boldsymbol{f}$ (Eq. 37). The posture control emerges in a task-consistent manner, where the *posture servo* block represents a neuromuscular-based control law. This can be defined as the gradient of a muscle effort potential plus a damping term (Eq. 39). The sum of the task and posture terms is applied to the skeletal plant as a control input, $\boldsymbol{\Gamma}$ (Eq. 36)

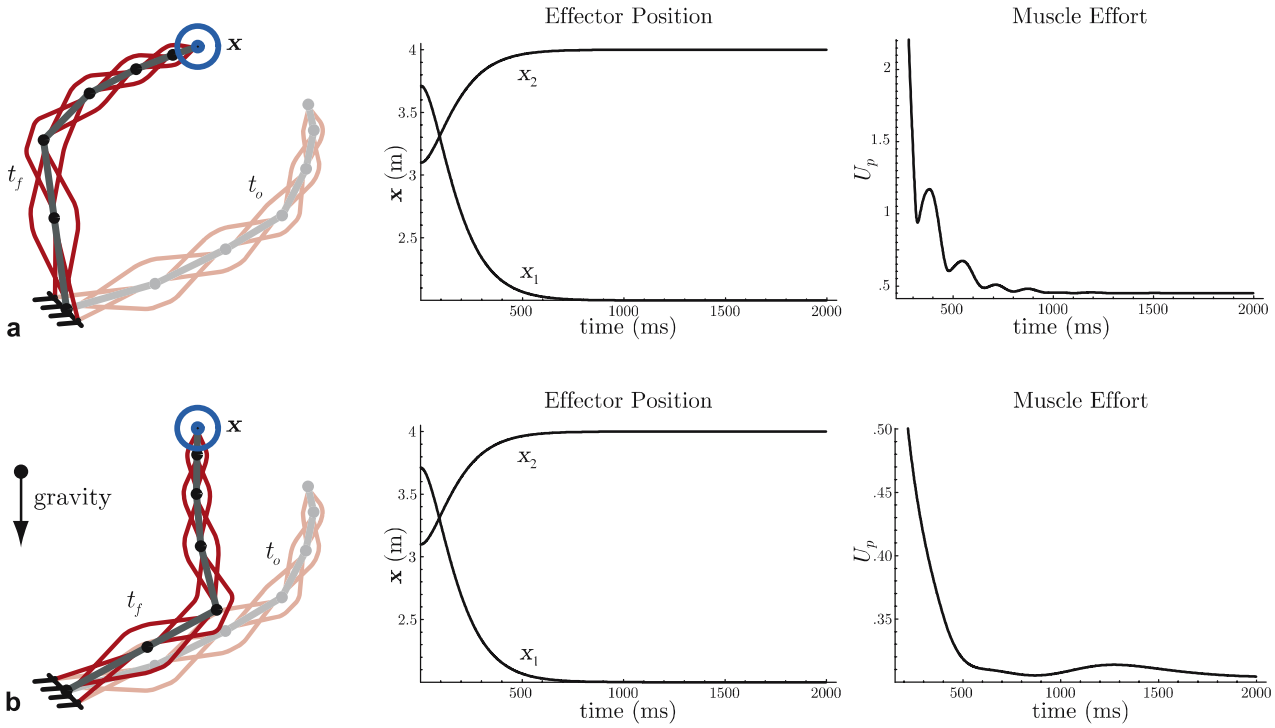


Fig. 7. A muscle-driven redundant chain controlled by the system of Fig. 6. The case where the distal muscles are stronger than the proximal ones is illustrated in (**a**). The controller drives the musculoskeletal plant to the target (task) from t_o to t_f in a manner that minimizes the effort ($K_x = 100$, $K_v = 20$, $K_p = .3$, $K_d = 3$). The case where the proximal muscles are stronger than the distal ones is illustrated in (**b**). Again, the controller drives the musculoskeletal plant to the target in a manner that minimizes the effort ($K_x = 100$, $K_v = 20$, $K_p = .4$, $K_d = 4$)

Different behaviors autonomously emerge from the controller based on these different physiological conditions.

The tight integration of our musculoskeletal model and task-level control framework is unique in that it poses the control objective for the musculoskeletal plant in terms of a concise set of task coordinates rather than the much larger set of generalized coordinates. Additionally, while the musculoskeletal plant can be driven by individual neural excitations, as is conventionally done, its integration with the task-level control framework allows it to be controlled through a unique task/posture decomposition. In this way, the posture controller encodes the properties of the musculoskeletal plant in an efficient manner, using a potential, without the need for specifying a large set of control inputs (neural excitations).

This approach also offers advantages over other approaches. For example, a standard approach like dynamic optimization exhibits a significant computational burden, since a large input space of neural excitations must be searched to drive the musculoskeletal plant over a fixed time interval. Additionally, the control is inherently open-loop as the feed-forward path of the musculoskeletal plant is repeatedly cycled in order to generate the control inputs. This prohibits the simulation from being interactive and responding in a robust manner to disturbances.

6 Simulation architecture

A simulation environment has been developed for simulating musculoskeletal dynamics and task-level control. This has been built upon a simulation and active interfaces framework (SAI), which is comprised of a set of libraries developed to perform interactive simulation of complex robotic systems [14].

6.1 Core simulation environment

At the core of SAI is a multi-body dynamics engine for simulating kinematic chains. Additionally, collisions and contact between objects are simulated. A haptics interface library allows for interaction with the system using a force-feedback device. Outside of the core dynamics and interaction capabilities, most of an SAI's infrastructure is devoted to algorithms for task-level planning and control.

The multi-body dynamics component of SAI focuses on recursive algorithms [1, 7, 20] for computing the dynamics and control of n-joint branching, redundant robotic mechanisms with m operational degrees of freedom. The complexity of these algorithms scales according to $\mathcal{O}(nm+m^3)$ as compared to $\mathcal{O}(n^3+m^3)$ for many existing symbolic algorithms. Typically, m is small compared to n, which implies near-linear scaling, $\mathcal{O}(n)$. A much more detailed description of the multi-body dynamics algorithms used in SAI is given [1].

The collision-handling capabilities of SAI implement an impulse-based approach for modeling rigid contact and collision. Objects in the simulated world are modeled as the union of convex polyhedra. A contact space is defined to describe the contact state between any two polyhedra with a finite number of contact points. When a contact or collision occurs, a Jacobian and operational space mass matrix, defined with respect to the contact space, are computed. In this approach, only the true degrees of freedom of the multi-body system are modeled, providing added efficiency by eliminating the internal constraints of the system. The collision model employs a standard coefficient of restitution relation as the constraint on velocities before and after collision. A much more detailed description of the collision approach used in SAI, as well as its benefits over standard approaches, is given in [24].

While collision capability is useful in the general application of physics-based simulation, we have not specifically exploited it in our task-level musculoskeletal control approach. As a result, we have only provided a brief summary of this core simulation capability and have not discussed it within the specific context of task-level musculoskeletal control. Future applications of our musculoskeletal control approach may exploit the collision and contact capabilities of SAI. These capabilities would be particularly relevant for gait and contact-force controllers, both of which would require force-sensing information in the feedback loop. Figure 8 depicts simulations in an SAI illustrating its handling of multi-body dynamics as well as collision.

6.2 Musculoskeletal simulation environment

The core SAI environment has been extended to incorporate the simulation of muscle-driven systems. Characters can be specified with a set of musculotendon actuators, modeled as described in Sect. 3. The environment allows feed-forward operation, in which control inputs are sent to each of the muscles, as well as an implementation of the task-level control methodology of Sect. 5. The object-oriented design employed provides a natural mechanism for connecting the musculoskeletal plant to task-level controllers. Appendix A provides a detailed description of the class design.

The addition of muscle simulation capabilities to SAI is illustrated in Fig. 9, which depicts a sequence from a task-level control simulation. The model shown in the sequence consists of 30 joints and 144 muscles. Musculoskeletal data used in this model has been derived from SIMM models [4]. The hands are defined as task-control points that can be moved interactively by the user. A balancing task constraint is imposed as well.

Because the control of the human characters in these simulations is robust and executed in real-time (or near real-time), the user can interact with the simulation and apply external disturbances to the characters. Figure 10

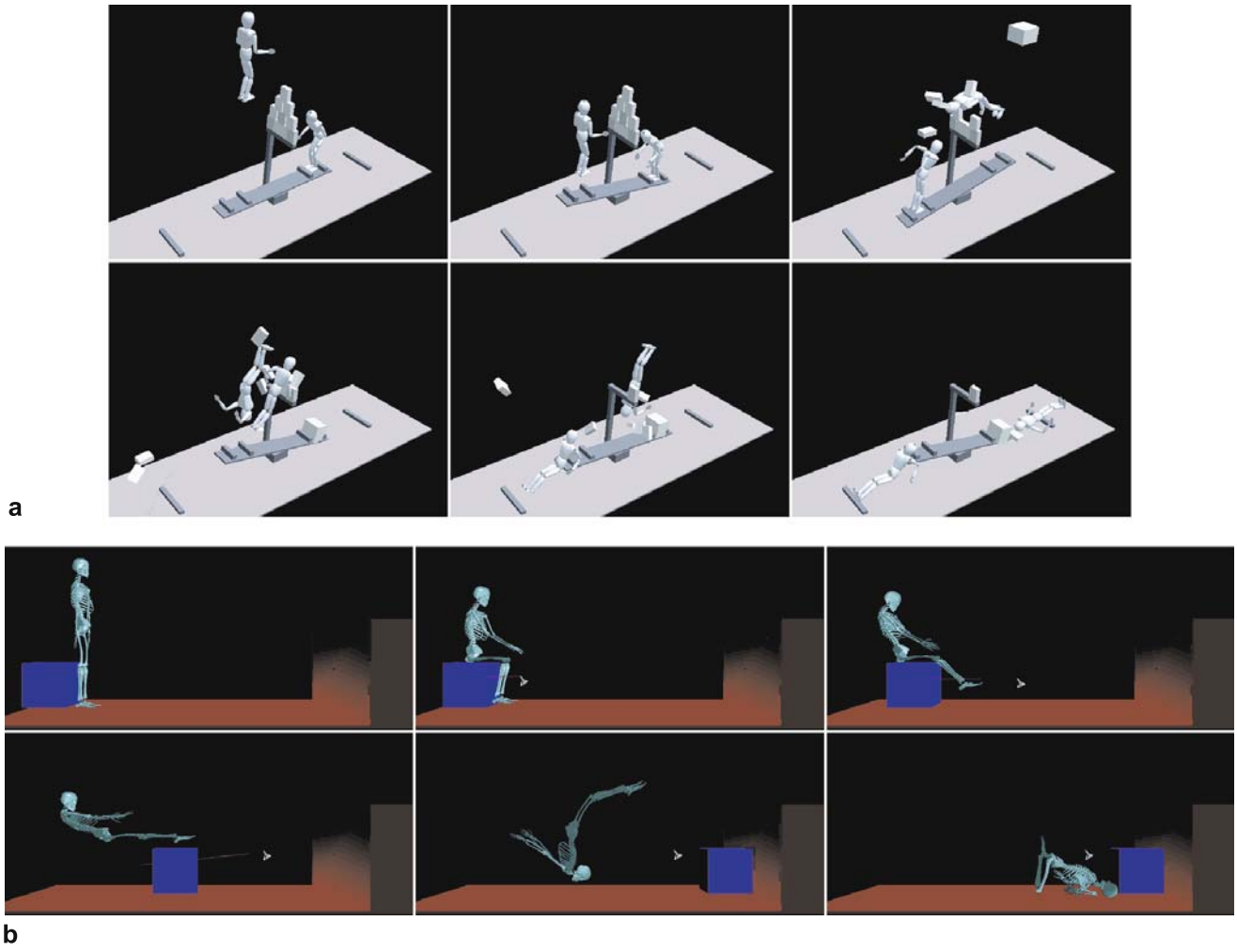


Fig. 8. Sequences illustrating core simulation capabilities of SAIs. Virtual acrobats interacting in a dynamic environment subject to realistic physical constraints (**a**). This SAI simulation sequence illustrates real-time multi-body dynamics and collision response (simulation sequence created by F. Conti). Interaction with a virtual skeleton in an SAI (**b**). The skeleton is commanded to sit on the block. The user then pulls on the block with a virtual force, causing the skeleton to summersault (simulation sequence created by V. De Sapio)

shows an example of disturbance forces interactively applied by the user. The character's controller applies compensation in response to the disturbance, resulting in stable behavior of the character's dynamics.

7 Conclusion and future work

We have presented a framework for the task-level control of muscle-driven human motion. This framework integrates a musculoskeletal model and a task-level control approach, allowing the control objective for the musculoskeletal plant to be stated in terms of a natural set of task coordinates. The unique task/posture decomposition implements a posture controller, which efficiently encodes musculoskeletal properties into a posture potential. The feedback nature of this framework inherently provides stable response to disturbances and external interactions, as opposed to feed-forward approaches like dynamic optimization.

A simulation environment that implements our task-level musculoskeletal control framework has also been presented. This environment is built upon an efficient multi-body dynamics engine. The overall environment provides a user-friendly interface for producing physically realistic task-driven human-like motions. We believe that such a system is useful in the study and synthesis of human-like motion for computer graphics applications as well as the synthesis of human-like motion control for anthropomorphic robotic systems.

The current focus of our work has been on whole-body postural control rather than walking. There are gait controllers currently being tested for future use. Regarding

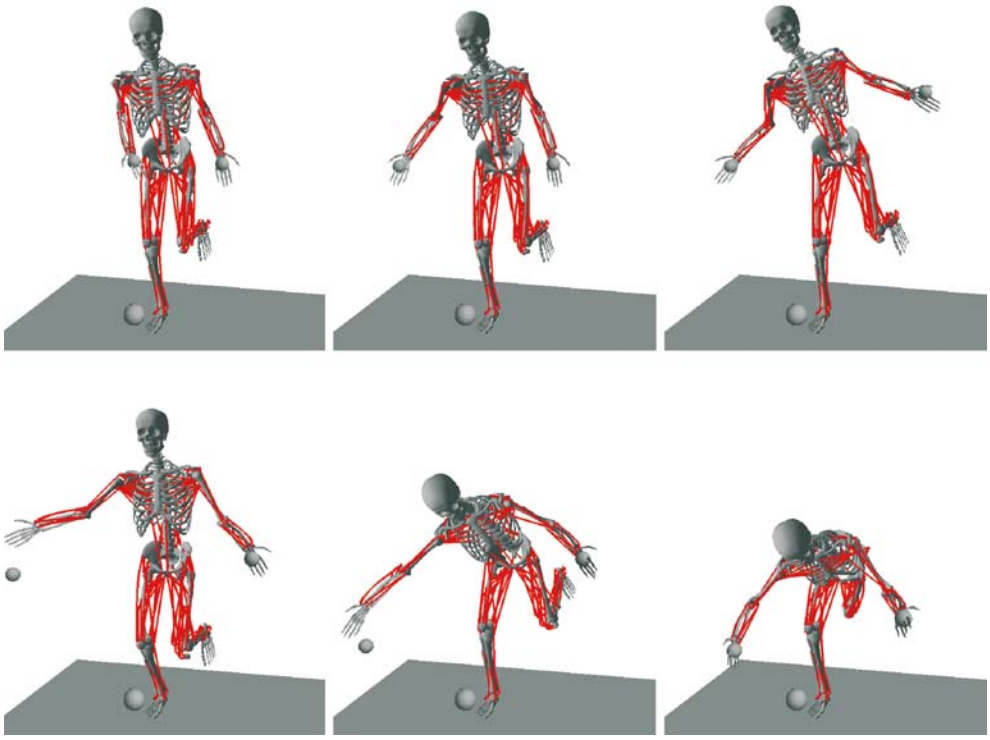


Fig. 9. Task-level control of a simulated human in a dynamic setting. While the environment allows feed-forward operation in which control inputs are sent to each of the 144 muscles, a more efficient task-level control is used in this interactive sequence. Users can interactively move the control points, defined at the wrists, while the simulated human responds

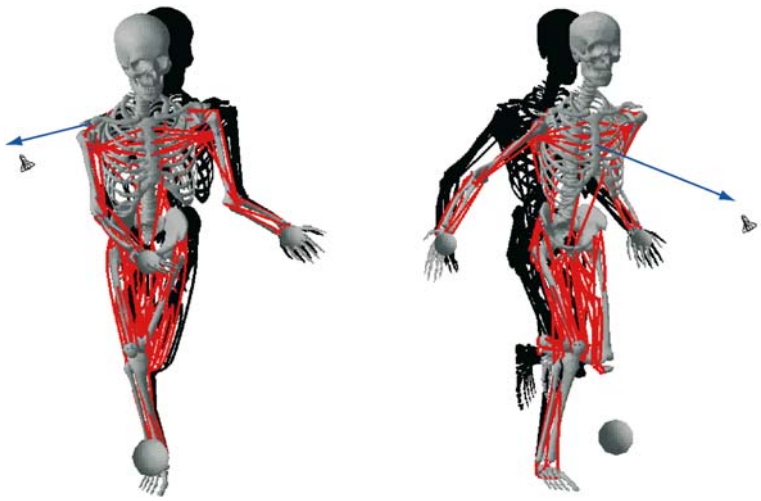


Fig. 10. Response of a human character to disturbance forces (vectors) interactively applied by the user. The feedback controller applies compensation resulting in stable behavior of the system and a return to equilibrium. This is not possible with feed-forward simulation approaches that generate open-loop controls

the muscle models, while the current system simulates full muscle dynamics in the feed-forward path, a stiff tendon muscle model is employed for control purposes. This simplifies the control problem but neglects some important dynamical properties of muscle. Plans for improvement of the system include the implementation of other muscle control routines. This will include an operational space version of the computed muscle control method [28]. Such an approach involves performing a task-constrained minimization of muscular effort defined in terms of muscle activations. The conceptual framework for this approach has been developed and an implementation is planned.

The implementation of constraint enforcement capabilities is planned as well. This step will be useful for modeling closed kinematic chains in addition to chains that involve constraints between the generalized coordinates. Such constraints are particularly relevant in the case of the human shoulder complex and the knee joint. A control methodology has been developed to address such cases [2]. This methodology makes use of a unique operational space decomposition, which generates sepa-

rate structures for task-motion control and constraint-force control.

The muscle path generation facilities of the SAI musculoskeletal environment do not include wrapping surfaces, which are useful for modeling complex muscle routing. SIMM includes these facilities and has been used to augment the capabilities of SAI in this area. Consequently, an implementation of wrapping surfaces would be a particularly useful addition to a future version of the SAI environment.

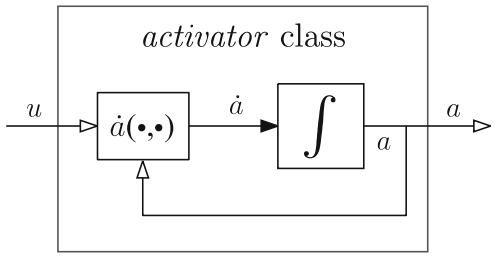
Acknowledgement The contributions of Luis Sentis, Irena Paschenko, François Conti, Diego Ruspini, and Kyong-Sok Chang to the SAI simulation environment are gratefully acknowledged. The authors would also like to thank Katherine Holzbaur for providing an enhanced model of the human shoulder complex, which was implemented in some of our work.

A musculoskeletal class design

Skeletal dynamics has been modeled in the SAI environment using a set of *link*, *joint*, and *robot* classes. At the top of the class hierarchy is a *world* class that contains a list of pointers to any number of *robot* objects. Each robot can be loaded individually into the world by the user. An XML file structure is used for storing geometric, kinematic, and inertial properties for each robot.

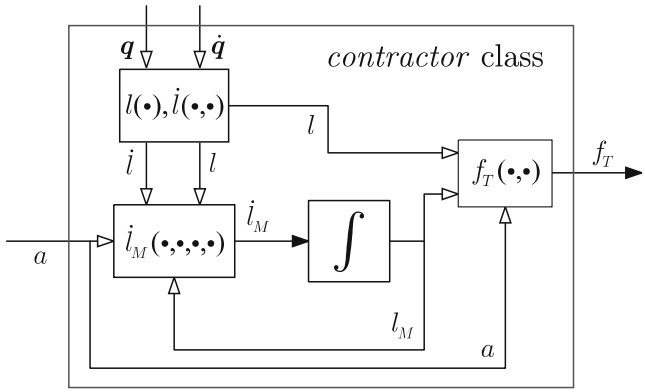
Muscle mechanics have been modeled using a class hierarchy consisting of an *activator* class, a *contractor* class, a *muscle* class, and a *muscle system* class. Collectively, the classes are used to define a system of controllable musculotendon actuators. Each robot object points to a muscle system object. An XML file structure is used for storing properties for the muscles that are contained in each muscle system. A set of structures are used for storing muscle and tendon lengths, velocities, and forces, as well as activation properties, contraction properties, and path data. The *path* structure contains a list of attachment points and a list of pointers to corresponding attachment links. There are also linear and cubic spline interpolation functions. The linear interpolation function is used for interpolating over the muscle force-length data. The spline interpolation function employs a Catmull-Rom spline algorithm to generate muscle paths (Fig. 11).

Fig. 11. Muscle paths are generated as Catmull-Rom splines through muscle origin, insertion, and via points



Inputs : *neural excitation* - u
Ouputs : *activation* - a
States : *activation* - a
Other : *activation rate* - $\dot{a}$

Fig. 12. The *activator* class state model. Activation, a, is the system state (and output) responding to neural input



Inputs : *activation* - a
: *configuration* - $\boldsymbol{q}, \dot{\boldsymbol{q}}$
Ouputs : *force* - f_T
States : *muscle length* - l_M
Other :
- *muscle/tendon lengths* - l, l_T
- *muscle/tendon velocities* - $\dot{l}, \dot{l}_M, \dot{l}_T$
- *active and passive forces* - f_A, f_P

Fig. 13. The *contractor* class state model. Muscle length, l_M, is the system state responding to activation input. The output is the tendon force, f_T

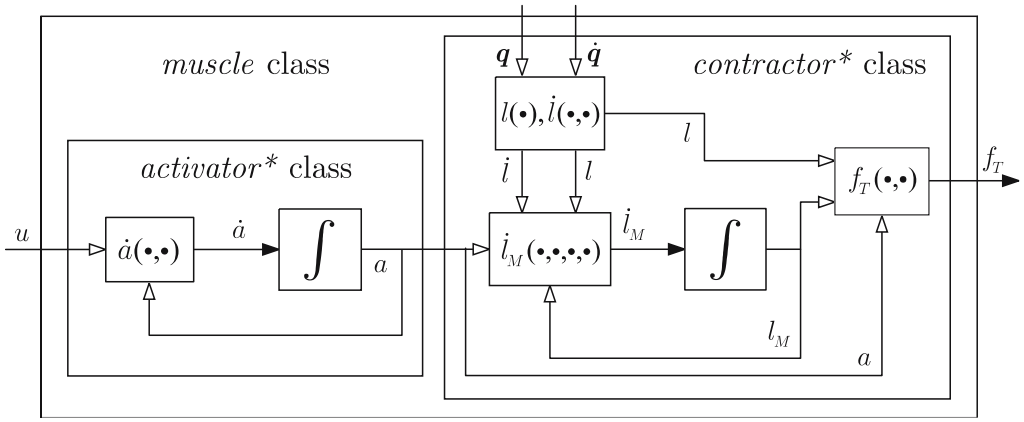


Fig. 14. The *muscle* class model consisting of pointers to a contractor and an activator object. A muscle object coordinates activation and muscle dynamics for a given muscle-tendon unit.

The activator class encapsulates muscle activation dynamics. Each object contains activation property data specific to an individual muscle. Based on a neural input, u, muscle activation, a, is computed at each time-step. Figure 12 depicts the block diagram for this class.

The contractor class encapsulates musculotendon contraction dynamics. Each object contains musculotendon path data and contraction properties data specific to an individual muscle-tendon pair. The overall musculotendon path length, l, is computed given the path data and the current configuration of the skeleton, $\boldsymbol{q}$. Based on an activation input, a, the force generated in the tendon, f_T, as well as active and passive forces generated in the muscle, f_A and f_P, are computed at each time-step. Figure 13 depicts the block diagram for this class.

The muscle class integrates the activator and contractor classes. It contains a pointer to an activator object and contractor object and thus represents a single musculotendon actuator (motor nucleus and associated muscle-tendon body). Methods are provided to interface to the activator and contractor objects. Figure 14 depicts the block diagram for this class.

The muscle system class is composed of a vector of pointers to individual muscle objects. There are methods to coordinate the initialization and update of individual muscle objects. Additionally, at each update, system-wide quantities are computed. These include the muscle Jacobian, $\boldsymbol{L}$, and the vector of joint torques, $\boldsymbol{\Gamma}$. The muscle Jacobian is computed given the path data and the current configuration of the skeleton.

The set of classes previously described simulate feed-forward musculoskeletal dynamics. A set of controls (neural excitations) can be specified as input to drive the feed-forward dynamics. In practice, these controls would be produced by a dynamic optimization routine [22] or in closed-loop by a feedback control routine [28]. To implement the feedback control structure described in Sect. 5, a set of classes have been implemented that employ the stiff tendon assumption described earlier. These are simplifications of the muscle classes already described and are used specifically for task-level musculoskeletal control. At each update, additional system-wide quantities are computed. These include the muscle force-activation gain matrix, $\boldsymbol{K}_f$ and the muscle torque-activation gain matrix, $\boldsymbol{K}_\Gamma$, as well as the muscle-induced torque boundaries Γ_{B_i}. Various muscle effort measures, like the ones described in Sect. 5.2, can be computed from these system-wide quantities.

References

1. Chang K-S, Khatib O (2000) Operational space dynamics: efficient algorithms for modeling and control of branching mechanisms. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation 1:850–856
2. De Sapio V, Khatib O (2005) Operational space control of multibody systems with explicit holonomic constraints. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (in press)
3. Delp SL, Loan JP (1995) A software system to develop and analyze models of musculoskeletal structures. Comput Biol Med 25:21–34
4. Delp SL, Loan JP (2000) A computational framework for simulating and analyzing human and animal movement. IEEE Comput Sci Eng 2(5):46–55
5. Faloutsos P, van de Panne M, Terzopoulos D (2003) Autonomous reactive control for simulated humanoids. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation 1:917–924
6. Faloutsos P, van de Panne M, Terzopoulos D (2001) Composable controllers for physics-based character animation. In: Proceedings of the 2001 SIGGRAPH Conference, pp 251–260
7. Featherstone R (1987) Robot dynamics algorithms. Kluwer, Boston
8. Hase K, Miyashita K, Ok S, Arakawa Y (2003) Human gait simulation with a neuromusculoskeletal model and evolutionary computation. J Visual Comput Anim 14(2):73–92
9. Hollars M, Rosenthal D, Sherman M (1991) SD/Fast users manual. Technical report, Symbolic Dynamics Incorporated
10. Ijspeert A, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation 2:1398–1403

11. Kandel ER, Schwartz JH, Jessell TM (2000) Principles of neural science, 4th edn. McGraw-Hill, New York
12. Khatib O (1987) A unified approach to motion and force control of robot manipulators: the operational space formulation. Int J Robot Res 3(1):43–53
13. Khatib (1995) Inertial properties in robotic manipulation: An object level framework. Int J Robot Res 14(1):19–36
14. Khatib O, Brock O, Chang K-S, Conti F, Ruspini D, Sentis L (2002) Robotics and interactive simulation. Commun ACM 45(3):46–51
15. Khatib O, Sentis L, Park J, Warren J (2004) Whole body dynamic behavior and control of human-like robots. Int J Humanoid Robot 1(1):29–43
16. Khatib O, Warren J, De Sapio V, Sentis L (2004) Human-like motion from physiologically-based potential energies. In: Lenarčič J, Galletti C (eds) On advances in robot kinematics. Kluwer, Boston, pp 149–163
17. Komura T, Shinagawa Y, Kunii TL (1999) Calculation and visualization of the dynamic ability of the human body. J Visual Comput Anim 10(2):57–78
18. Komura T, Shinagawa Y, Kunii TL (2000) Creating and retargetting motion by the musculoskeletal human body model. Vis Comput 16(5):254–270
19. Kuroki Y, Blank B, Mikami T, Mayeux P, Miyamoto A, Playter R, Nagasaka K, Raibert M, Nagano M, Yamaguchi J (2003) Motion creating system for a small biped entertainment robot. In: Proceedings of the 2003 International Conference on Intelligent Robots and Systems 2:1394–1399
20. Lilly KW (1992) Efficient dynamics simulation of robotic mechanisms. Kluwer, Boston
21. McMahon TA (1984) Muscles, reflexes, and locomotion. Princeton University Press, Princeton
22. Pandy MG, Anderson FC, Hull DG (1992) A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. J Biomech Eng 114:450–460
23. Rasmussen J, Damsgaard M, Surma E, Christensen S, de Zee M (2003) Designing a general software system for musculoskeletal analysis. In: Proceedings of the International Symposium on Computer Simulation in Biomechanics
24. Ruspini D, Khatib O (2000) A framework for multi-contact multi-body dynamic simulation and haptic display. In: Proceedings of the 2000 International Conference on Intelligent Robots and Systems 2:1322–1327
25. Sakagami Y, Watanabe R, Aoyama C, Matsunaga S, Higaki N, Fujimura K (2002) The intelligent asimo: system overview and integration. In: Proceedings of the 2002 International Conference on Intelligent Robots and Systems 3:2478–2483
26. Schutte LM (1992) Using musculsokeletal models to explore strategies for improving performance in electrical stimulation-induced leg cycle ergometry. Dissertation, Stanford University
27. Thelen DG, Anderson FC (2001) An operational space tracking algorithm for generating dynamic simulations of movement. In: Proceedings of the 5th International Symposium on Computer Methods in Biomechanics and Biomedical Engineering
28. Thelen DG, Anderson FC, Delp SL (2003) Generating dynamic simulations of movement using computed muscle control. J Biomech 36:321–328
29. Yamane K (2004) Simulating and generating motions of human figures. Springer, Berlin Heidelberg New York
30. Zajac FE (1989) Critical reviews in biomedical engineering. In: Bourne JR (ed) Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. CRC Press, Boca Raton 17(4)359–411
31. Zajac FE (1993) Muscle coordination of movement: a perspective. J Biomech 26:109–124




VINCENT DE SAPIO is a Ph.D. student in the Artificial Intelligence Laboratory at Stanford University and a senior member of the technical staff at Sandia National Laboratories. He received a B.S. degree from Rensselaer Polytechnic Institute in 1990 and an M.S. degree from Stanford University in 1994, both in Mechanical Engineering. He is currently investigating task-level control within the context of biomechanical systems. This involves the implementation of neuromuscular models in conjunction with a task/posture control decomposition. The goal is to emulate aspects of human motor control in robotic systems.

JAMES WARREN is a Ph.D. student in the Artificial Intelligence Laboratory at Stanford University. He received his B.S. degree in Mathematics from Texas A&M University in 1996 and his M.S. degree in Scientific Computing and Computational Mathematics from Stanford University in 2000. His research is in identifying human motion behaviors from motion capture data and applying these strategies for humanoid robotic control.

OUSSAMA KHATIB is Professor of Computer Science at Stanford University. He received his Ph.D. in 1980 from Sup'Aero, Toulouse, France. His current research is in human-centered robotics, human motion synthesis, humanoid robotics, dynamic simulation, haptic interfaces, and human-friendly robot design. This builds upon a large body of work pursued over the past 25 years and published in over 200 contributions in the field. Professor Khatib was Program Chair of ICRA2000 and editor of *The Robotics Review*. He served as director of the Stanford Computer Forum and is currently president of the International Foundation of Robotics Research and editor of *Springer Tracts in Advanced Robotics*. Professor Khatib is an IEEE Fellow, a Distinguished Lecturer of IEEE, and a recipient of the JARA Award.

SCOTT DELP is Professor and Chair of the Bioengineering Department at Stanford University. After receiving a Ph.D. degree from Stanford University in 1990, he joined the faculty of Northwestern University, where he was jointly appointed in the Schools of Engineering and Medicine. He returned to Stanford in 1999 in the departments of Mechanical Engineering and Bioengineering. His work draws on computational mechanics, biomedical imaging, and neuromuscular biology to study human movement. He has received numerous awards, including a National Young Investigator Award from the National Science Foundation and Faculty Fellowships from the Baxter Foundation and Powell Foundation. He is a Fellow of the American Institute of Medical and Biological Engineering.