Biomechanical Simulation and Control of Hands and Tendinous Systems

Prashant Sachdeva[1]* Shinjiro Sueda[2]* Susanne Bradley[1] Mikhail Fain[1] Dinesh K. Pai[1]
[1]University of British Columbia [2]California Polytechnic State University

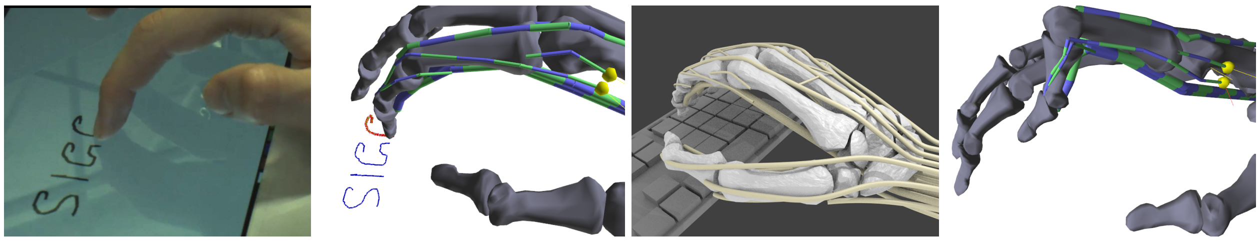


Figure 1: *Our biomechanical simulation and control framework can model the human hand performing tasks such as writing (a-b), and typing on a keyboard (c). We can also simulate clinical conditions such as boutonniére deformity (d) by cutting a tendon insertion.*

Abstract

The tendons of the hand and other biomechanical systems form a complex network of sheaths, pulleys, and branches. By modeling these anatomical structures, we obtain realistic simulations of coordination and dynamics that were previously not possible. First, we introduce Eulerian-on-Lagrangian discretization of tendon strands, with a new selective quasistatic formulation that eliminates unnecessary degrees of freedom in the longitudinal direction, while maintaining the dynamic behavior in transverse directions. This formulation also allows us to take larger time steps. Second, we introduce two control methods for biomechanical systems: first, a general-purpose learning-based approach requiring no previous system knowledge, and a second approach using data extracted from the simulator. We use various examples to compare the performance of these controllers.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Combined

Keywords: muscles, tendons, hands, physically based simulation, constrained strands, Lagrangian mechanics

1 Introduction

Although simulations of hands and grasp have consistently received attention in the graphics community, modeling and simulating the dynamics of the complex tendon network of the hand has remained relatively unexplored. Much of the previous simulation techniques for the hand have been based on rigid links with either joint torques [Pollard and Zordan 2005; Kry and Pai 2006; Liu 2009] or line-of-force muscles [Albrecht et al. 2003; Tsang et al. 2005; Johnson et al. 2009]. In the real hand, however, the motion of the digital phalanges are driven by muscles originating in the forearm acting through tendons passing through a complex network of sheaths and pulleys [Valero-Cuevas et al. 2007]. This design has an important effect on the compliance and coupling of the fingers.

In our approach, we model each tendon as a physical primitive rather than using joint torques or moment arms. This is advantageous because it allows us to properly model the complexity of the tendon network, which we believe is important for obtaining realistic motions. As an added bonus, the biomechanical modeling of tendons and ligaments can give us proper joint coupling for free. As a simple example, let us consider the coordinated motion of the joints, shown in Fig. 7. The two distal joints of the finger (PIP and DIP) flex and extend in a coordinated fashion *not* because of the synchronized activation signals computed by the brain, but because of the arrangement of tendons and ligaments in the finger. In our simulator, pulling on a single tendon (flexor digitorum profundus) achieves this result, whereas with a torque-based approach, we would need to manually coordinate the torques at these two joints. Another example is in the coupling between the extensor tendons in the back of the hand. Although this lack of complete independence between the fingers is partly due to the neural control, mechanical coupling has been hypothesized as having a significant role [Lang and Schieber 2004]. We are also able to simulate hand deformities, which have applications not only in surgical planning and medicine, but also in computer graphics (Figs. 1(d) and 8). Some virtual character designs are based on deformities or injuries, and the ability to procedurally produce anatomically-based abnormal characters may prove useful, since obtaining real-world data of such characters can be a challenge.

We also introduce two methods for control of biomechanical systems. The first is a general-purpose learning method requiring no previous knowledge of the inner workings of the system and is suitable for control of any "black box" simulator. The second controller extracts controller parameters from the internals of the simulator.

Contributions We address three main issues that are especially important for biomechanical simulation. First, by applying the Eulerian-on-Lagrangian discretization of the tendon strand [Sueda et al. 2011], we greatly simplify the contact handling between tendons and bones. Second, we develop a new formulation to deal with highly stiff strands, by assuming that strain and stress propagate instantaneously through the strand, allowing us to use large time steps even for stiff tendons (§3.2). Third, we introduce and assess two methods for control of these systems (§4.2, §4.3).

*Equal contributions.

2 Related Work

Our method is most closely related to the biomechanical strand model introduced by Sueda et al. [2008]. We extend the constrained strand framework of Sueda et al. [2011] to model tendons, which gives us more robust handling of constraints between tendons and bones. Without the constrained strand framework, even relatively simple scenarios, such as the flexor pulleys shown in Fig. 3b, are difficult to model, since the discretization of the tendon strand cannot always match the discretization of the pulley constraints. As far as we know, all previous models of tendon elastodynamics, including [Sueda et al. 2008], suffer from this difficulty. The regions between the pulleys in Fig. 3b are also difficult to model, since constraining the degrees of freedom (DoFs) in these areas may rigidify the strand around the surrounding constraints. With the new extensions discussed in this paper, we are able to efficiently model the tendons of the hand including branching and contact with the bones at speeds comparable to previous approaches for tendon elastodynamics.

2.1 Musculotendon Simulator

Previous works on the hand in computer graphics have focused on geometry acquisition and retargeting of the whole hand [Kurihara and Miyata 2004; Li et al. 2007; Huang et al. 2011], musculotendon simulation based on kinematic muscle paths and moment arms [Albrecht et al. 2003; Tsang et al. 2005], and grasping and interaction with the environment [ElKoura and Singh 2003; Kry and Pai 2006; Pollard and Zordan 2005; Liu 2008; Liu 2009; Zhao et al. 2013; Wang et al. 2013]. In our work, we focus on the movement of the fingers and not of the hand and the forearm. Since the bones are represented as rigid bodies in our simulator, it should not be difficult to append the tendinous hand from our simulator to an arm composed of rigid body links, or to apply the control strategies for grasping presented in these previous works.

Biomechanical simulators have been used successfully in computer graphics on many other parts of the body, such as the face [Sifakis et al. 2005], the neck [Lee and Terzopoulos 2006], the upper body [Lee et al. 2009], and the lower body [Lee et al. 2014]. These, and other simulators from the field of biomechanics, cannot fully handle the complex routing constraints present in the hand, and the dynamic coupling of musculotendons and bones. In line-of-force models [Garner and Pandy 2000; Delp et al. 2007; Damsgaard et al. 2006; Valero-Cuevas et al. 2007; Johnson et al. 2009], muscles and tendons are simulated quasistatically as kinematic paths, with no mass and inertia, and do not fully account for the shape of the biomechanical structures. More importantly for hand simulation, handling of routing constraints, including branching and kinematic loops, is difficult with these methods. Simulators based on solid mechanics models, such as spline volumes [Ng-Thow-Hing 2001], finite volume method [Teran et al. 2003; Teran et al. 2005], finite element method, [Chen and Zeltzer 1992; Zhu et al. 1998; Sifakis et al. 2005; Blemker and Delp 2005; Kaufman et al. 2010], or even Eulerian solids [Fan et al. 2014] are also not ideal for hand simulation, since the musculotendons of the hand are thin and anisotropic, and would require many disproportionately small elements. Various dynamic models [Spillmann and Teschner 2008; Bergou et al. 2008; Bergou et al. 2010] from the computer graphics community could potentially be used for musculotendon simulation, but these models were designed for use in free-floating configurations, and do not work well in the highly-constraining situations present in the hand. A hybrid approach has also been used, where the muscles are abstracted as piecewise linear actuators that drive the volumetric muscle based on solid-mechanics [Teran et al. 2005; Lee et al. 2009]. Our method could be used to replace the underlying linear muscle model of these approaches.

2.2 Musculotendon Controller

Our work on the controller focuses on the trajectory tracking problem: finding input u_t such that the plant trajectory is as close as possible to the desired trajectory. An important special case is the reaching problem, where the desired trajectory is constant. Trajectory tracking is crucial for learning complex motions, such as grasping or obstacle avoidance. The optimization algorithms in this area include Policy Improvement with Path Integrals (PI^2) [Rombokas et al. 2012], Covariance Matrix Adaptation (CMA) [Geijtenbeek et al. 2013; Wang et al. 2012], and others [Mordatch et al. 2012; Andrews and Kry 2013].

Trajectory tracking of linear systems is a well-studied problem with an analytical solution, but the methods for controlling non-linear systems are applicable only in restricted settings. For controlling complex non-linear plants which do not conform to those settings, the hierarchical control approach is typically used [Zhang et al. 2011; Liu 2009; Hou et al. 2007; Fortney and Tweed 2012; Geijtenbeek et al. 2013]. This approach involves solving a simpler control problem on a kinematic level first, and then using a low-level controller to compute the motor controls from the kinematic-level controls.

Previous work on trajectory tracking of anatomically correct tendon-driven human hand models includes controlling the Anatomically Correct Testbed (ACT) hand [Malhotra et al. 2012; Deshpande et al. 2013] and a hand modeled with Lagrangian strands [Sueda et al. 2008]. The motor controls were computed by solving a linear constrained least-squares problem, with the linearization derived directly from the simulator [Sueda et al. 2008] or learned from experience [Malhotra et al. 2012].

3 Simulation Framework

One of the main challenges in building a musculotendon simulator is constraint handling. A tendon moves freely in the axial direction but is constrained throughout its length by a series of sheaths and pulleys. Fig. 2 shows the complexity of the tendon network of the finger. To handle this complexity, we use the *Eulerian-on-Lagrangian strands* approach of Sueda et al. [2011].

3.1 Eulerian-on-Lagrangian Strands

A strand is represented as a series of nodes, each of which contains both *Lagrangian* and *Eulerian* coordinates: $(\boldsymbol{x}_i, s_i)$. The Lagrangian coordinates of a node, $\boldsymbol{x}_i$, are the world position of the node, and the Eulerian coordinate, s_i, is the material coordinate of the strand at that node. See Fig. 3a for an illustration. The Lagrangian coordinates encode the geometric path of the strand, and the Eulerian coordinate encodes the actual strand material at the nodes.

This separation of the geometric path and the strand material is critical for a musculotendon simulation because it decouples the routing constraints from the dynamics. Consider, for example, the pulleys holding the flexor tendons in place (A-1 through A-5 in Fig. 3b). With a purely Lagrangian approach, constraining the tendon to go through these pulleys would require the tendon to be discretized very finely near the start and end of each of the pulleys. With the Eulerian-on-Lagrangian approach, we require just the right number of nodes, since we can place the nodes exactly at the start and end of the pulleys.

We take advantage of the fact that tendons are always in close contact with other anatomical structures, and we know a-priori where the routing constraints on the tendons are going to be. We therefore do not need a general formulation of contact and its associated cost

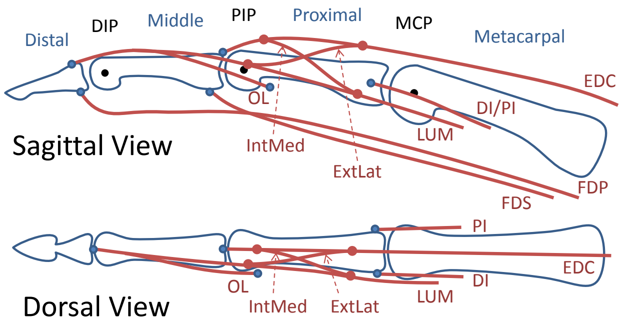


Figure 2: *Anatomy of the finger (of the right hand) showing the components modeled. From proximal to distal, the four bones of the finger are metacarpal, proximal, middle, and distal. They are connected by the metacarpophalangeal (MCP), proximal-interphalangeal (PIP), and distal-interphalangeal (DIP) joints. The three long tendons are extensor digitorum communis (EDC), flexor digitorum superficialis (FDS), and flexor digitorum profundus (FDP). These tendons originate from outside the hand, and are hence called "extrinsic." The three "intrinsic" musculotendons are the lumbrical (LUM), dorsal interosseus (DI) and palmar interosseus (PI). Each finger has a slightly different insertion arrangement of these three intrinsic musculotendons. We simplified the model by using the arrangement in the index finger for all fingers and omitting the symmetric components on the ulnar side. The oblique ligament (OL), which spans the PIP and DIP joints, helps synchronize these two joints when the FDP is pulled. The two crossing tendons, extrinsic lateral (ExtLat) and intrinsic medial (IntMed) transfer tension from extrinsic extensors and intrinsic extensors to the PIP and DIP. (Not present in all the examples.)*

in simulation and modeling time. Collisions need to be handled only by the Lagrangian part of the discretization, which determines the kinematic path of the strand; the Eulerian part is oblivious to collision detection and resolution. This is a major advantage for tendon simulation, since most of the movement is in the axial direction, which means that the Lagrangian part does not move in space too much, and is often completely stationary with respect to the skeleton.

3.2 Selective Quasistatics

Tendons are very stiff, making them challenging to simulate numerically. One approach for incorporating such stiff forces is to approximate them by hard constraints. Ideally, unilateral constraints are preferred, since tendons should not be able to push—i.e., the tendon strand should prevent stretch but allow compression. However, unilateral constraints are more costly to maintain than bilateral constraints, and so should be avoided as much as possible.

There are two problems with modeling a stiff tendon as a bilaterally constrained strand. First, as we stated earlier, bilateral constraints prevent strand compression, which means that they can push as well as pull. However, because of the design of our muscle model described in §3.3, muscles never push back on the tendons, and so this is not a serious drawback in all cases. Second, and more importantly, bilateral constraints can cause locking in some situations that are prevalent in hand simulation because of the complexity of the tendon network of the finger, which contains several branching and merging tendons. Modeling tendons as inextensible strands can cause this tendon network to turn into a reduced degrees-of-freedom (DoF) system. With our formulation, we can easily model these loops, as shown in Fig. 7.

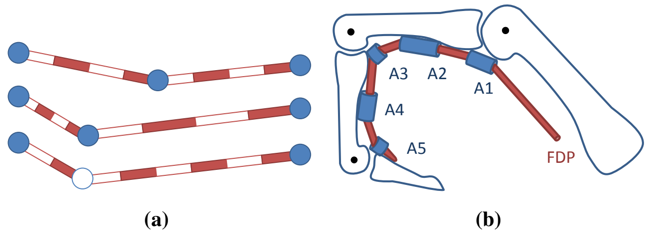


Figure 3: *(a) Illustration of Eulerian coordinates with a stretched elastic band. (top) In a purely Lagrangian simulator, the strand material, which can be visualized as the texture, is fixed to the nodes. (middle) If we move the middle node to the left, then the material is compressed in the left segment and stretched in the right segment with respect to the previous configuration. (bottom) If we relax the assumption that the material is fixed to the nodes (i.e., Eulerian node), then the material will start to flow from left to right. (b) Pulleys to hold the tendon in place.*

Our solution is to use bilateral constraints only when the origin of the strand is free. The high stiffness of these strands that do not form loops can be safely approximated by bilateral constraints. For those special cases where the strand forms a loop, we use a novel *selective quasistatic* discretization of the strand. The basic assumption of the selective quasistatic discretization is that the strain propagates instantaneously along the strand. This is similar to the idea proposed for discrete elastic rods by Bergou et al. [2008], where the twist of the strand was assumed to propagate instantaneously. Similarly, we propose to enforce all neighboring segments to have the same strain, eliminating the Eulerian coordinates from the equations of motion. Note the *selective* nature of the elimination—even if we eliminate the Eulerian DoFs from the equations of motion, the strand is still dynamic since its Lagrangian DoFs are still present.

Static condensation techniques, e.g., [DiMaio and Salcudean 2002], remove all internal degrees of freedom using a quasistatic assumption. With our selective quasistatic formulation, on the other hand, we gain the ability to insert Lagrangian DoFs without increasing the number of Eulerian DoFs—a flexibility that is useful for routing tendons around bones.

Another advantage of using selective quasistatics is improved conditioning, allowing larger time steps. As an illustration, imagine two consecutive nodes coming very close to each other at some finger posture (e.g., the small spacing between the A2 and A3 pulleys in Fig. 3b). Intuitively, without the quasistatic assumption, this would cause the Eulerian DoF of a node to affect only a small amount of the mass of the strand located around these nodes. With selective quasistatics, these small inertias are distributed to their surrounding Lagrangian DoFs, improving the condition number of the system matrix, thus allowing us to take larger time steps. The comparison of the largest time steps allowed as a function of the number of nodes is shown in Fig. 4. With quasistatic nodes, the maximum time step remains constant, whereas without the quasistatic assumption, the time step decreases rapidly with the number of nodes.

Consider a strand consisting of $n+1$ segments as shown in Fig. 5. Let node 0 denote the first node, and $n+1$ denote the last node. Using the quasistatic strain assumption, we want to eliminate s_1 through s_n by expressing them as a function of the Lagrangian DoFs $(\boldsymbol{x}_0, \cdots, \boldsymbol{x}_{n+1})$ and the first and last Eulerian DoFs (s_0, s_{n+1}). The derivation starts[1] with the computation of the strain of segment i between nodes i and $i+1$: $\varepsilon_i = \frac{l_i}{\Delta s_i} - 1$, where $\Delta s_i = s_{i+1} - s_i$,

[1]See the supplemental document for a detailed derivation.

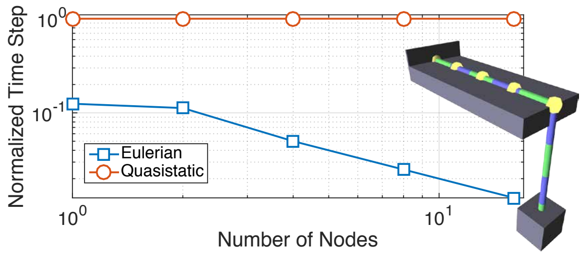


Figure 4: *An elastic strand stretching and sliding on a plank. The left end of the strand is fixed, and the right end is attached to the hanging box. The left-most node is a standard Lagrangian node; the remaining are Eulerian nodes that allow the strand material to slide through them. The Lagrangian coordinates of all the nodes are fixed to the plank. The plot shows a log-log comparison of maximum time steps (Δt) vs. the number of Eulerian nodes with and without selective quasistatics. With selective quasistatics (red), the maximum Δt remains constant (normalized to 1.0). Without selective quasistatics (blue), the maximum Δt decreases rapidly.*

$\Delta\boldsymbol{x}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$, and $l_i = \|\Delta\boldsymbol{x}_i\|$. These quantities are computed from the coordinates of the two nodes of the i^{th} segment. The quasistatic strain constraint implies that the strain values of all the segments are equal; i.e., $\varepsilon_0 = \cdots = \varepsilon_i = \varepsilon_{i+1} = \cdots = \varepsilon_n$. Using the expression for strain given above and rearranging, we can rewrite $\varepsilon_i = \varepsilon_{i+1}$ as

$$-l_{i+1}s_i + (l_i + l_{i+1})s_{i+1} - l_i s_{i+2} = 0. \tag{1}$$

This holds for all neighboring segments $i = 0, \cdots, n-1$. Assembling these into a linear system, we obtain a tridiagonal matrix equation $Ls = b$, where

$$L = \begin{pmatrix} l_0 + l_1 & -l_0 & & & \\ -l_2 & l_1 + l_2 & -l_1 & & \\ & & \ddots & & \\ & & -l_{n-1} & l_{n-2} + l_{n-1} & -l_{n-2} \\ & & & -l_n & l_{n-1} + l_n \end{pmatrix}$$
$$s = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n-1} \\ s_n \end{pmatrix}, \qquad b = \begin{pmatrix} l_1 s_0 \\ 0 \\ \vdots \\ 0 \\ l_{n-1} s_{n+1} \end{pmatrix}. \tag{2}$$

L is $(n \times n)$, s is $(n \times 1)$, and b is $(n \times 1)$. Solving this equation gives the Eulerian DoFs $(s_1, \ldots, s_n)$ that make the strain the same throughout the strand given the Lagrangian DoFs $(\boldsymbol{x}_0, \cdots, \boldsymbol{x}_{n+1})$ and the first and last Eulerian DoFs (s_0, s_{n+1}). In order to eliminate these DoFs, we also need the Jacobian matrix, which maps the velocities of the remaining DoFs to the velocities of the eliminated DoFs. Taking the time derivative of $s = L^{-1}b$ and using the identity for the derivative of the inverse, we arrive at

$$\begin{pmatrix} \dot{s}_1 \\ \vdots \\ \dot{s}_n \end{pmatrix} = \underbrace{-L^{-1}\Delta S \Delta X}_{J} \begin{pmatrix} \dot{\boldsymbol{x}}_0 \\ \vdots \\ \dot{\boldsymbol{x}}_{n+1} \end{pmatrix}. \tag{3}$$

(We assumed here that s_0 and s_{n+1} are fixed. It is also possible to relax this assumption.) The resulting $(n \times 3(n+2))$ matrix, J, is the Jacobian for the eliminated Eulerian coordinates. The blocks of

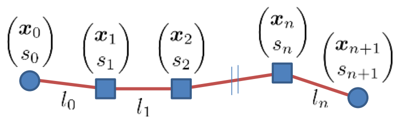


Figure 5: *If we assume that the strain is the same throughout the strand, we can eliminate the Eulerian coordinates s_1 through s_n given $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n+1}$, s_0, and s_{n+1}.*

J are composed of:

$$\Delta S = \begin{pmatrix} -\Delta s_1 & \Delta s_0 & & \\ & \ddots & & \\ & & -\Delta s_n & \Delta s_{n-1} \end{pmatrix},$$
$$\Delta X = \begin{pmatrix} -\Delta\bar{\boldsymbol{x}}_0^T & \Delta\bar{\boldsymbol{x}}_0^T & & \\ & \ddots & & \\ & & -\Delta\bar{\boldsymbol{x}}_n^T & \Delta\bar{\boldsymbol{x}}_n^T \end{pmatrix}, \tag{4}$$

where $\Delta\bar{\boldsymbol{x}} = \Delta\boldsymbol{x}/\|\Delta\boldsymbol{x}\|$. ΔS is $(n \times (n+1))$, and ΔX is $((n+1) \times 3(n+2))$. J is of size $(n \times 3(n+2))$ and is the Jacobian for the eliminated Eulerian coordinates. Although J is dense, it is relatively small, and since all of the matrices involved are banded, it is cheap to form.

This Jacobian matrix can then be used to eliminate the internal Eulerian DoFs from equations of motion (i.e., reduced coordinate approach), or it can be used to define the equality constraint matrix between the internal Eulerian DoFs and the remaining DoFs (i.e., maximal coordinate approach). In our implementation, we take the reduced approach to obtain a smaller but denser system matrix.

3.3 Muscle Model

Muscles are complex active materials with significant volumetric effects; practical volumetric muscle models have been recently developed (e.g., [Fan et al. 2014]). In the present context we propose a simpler lumped muscle model, which relates the force at the origin of the tendon to the tendon excursion (i.e., displacement of the tendon origin).

Such lumped models are widely used in biomechanics, but our model differs significantly from the standard Hill-Zajac model [Zajac 1989], and may be more useful in applications. In the standard model, the force exerted by a muscle is modeled as the sum of passive and active forces: $f = f_P + f_A$, usually depicted as the muscle's passive and active force-length (FL) curves. The passive FL curve is a monotonically increasing function, and the active FL curve is a concave function obtained from physiological experiments, starting with the classic experiments of A. V. Hill a century ago. When the muscle is activated, the active FL curve is scaled by the activation level of the muscle. The resulting total force, shown in Fig. 6(left), contains a region of negative stiffness, which means that as an active muscle is stretched, its force output decreases. Numerically, the negative slope manifests itself as instabilities in simulations, since any perturbation away from equilibrium is magnified by the negative force.

We therefore use a simple piecewise linear model that does not suffer from these difficulties; more detailed, non-linear constitutive models can also be used, depending on the application. We model the total muscle force as a piecewise linear function that is shifted to the left when activated. Fig. 6(middle & right) shows our muscle model. The passive FL curve is simply the unshifted curve. When the muscle is activated, it pulls on the tendon even at zero excursion

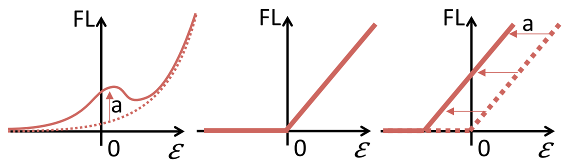


Figure 6: *(left) A typical FL curve of an activated muscle, depicting force as a function of tendon excursion. (middle) Piece-wise linear muscle model. The muscle exerts no force if the excursion is non-positive. The force increases linearly otherwise. (right) When the muscle is activated, the functions are shifted to the left.*

(i.e., in the isometric condition). Finally, because the function is always positive, the muscle never pushes back on the tendon.

Even though this model is simple and robust, it is also potentially more realistic than the standard model as a constitutive model of controlled muscles. The problem with the direct application of Hill's widely misunderstood FL curve is that it is *not* a constitutive model of active muscle; rather, it is a depiction of *maximum* force when *isometrically* activated at different lengths. It is now well known that muscle activation and stretch do not commute. Unlike the predictions of the standard model, it has been shown experimentally that if a muscle is activated first and then stretched, the resulting force does not decrease [Epstein and Herzog 1998]. Moreover, the behavior of entire muscles during stretch is mediated by a complex neural signal based on feedback from muscle proprioceptors and changes in fiber recruitment. There are very few studies of how muscles behave *in vivo*, especially in humans. A striking exception is the classic paper of Robinson, et al. [1969] which measured FL curves of active human eye muscles *in vivo*. Figure 2 in that paper is closer to ours than to the standard model. Our model needs further investigation, but there is physiological support for using it.

4 Control

In this section, we present our controller framework and discuss two methods used to learn the necessary control parameters. §4.2 describes a general learning method, to which we will refer henceforth as "black box control," that uses no prior system knowledge and is suitable for control of a "black box" simulator. We then present in §4.3 a method – denoted as "white box control" – for extracting the relevant parameters for control of a biomechanical system directly from the simulator. We compare these methods to explore how much we benefit from exploiting the inner workings of the system we are attempting to control, and to assess how a general-purpose learning method performs compared to a method with detailed prior system knowledge.

4.1 Controller Algorithm

Since our plant is non-linear and input-constrained, we employed a hierarchical control approach, modeling the plant as a simple unconstrained system on a higher (kinematic) level, and an input-constrained locally-linear system at the lower (muscle activation) level. For concreteness, we consider the index finger, but the method is more general.

At the high level, we model the fingertip motion as simple controlled dynamical system:

$$\dot{q}_{t+1} = \dot{q}_t + v_t \tag{5}$$

where q_t is a 3×1 task descriptor variable (in this case, the fingertip location) at time t, and v_t are the kinematic controls.

The kinematic controls v_t are defined by:

$$v_t := v_p + v_b. \tag{6}$$

Here $v_p = v_p(q_t, t)$ is the 3×1 *passive dynamics* term, the computation of which is described in §4.2; for a complete derivation of the equations of motion, we refer the reader to our supplementary material. v_p captures the significant nonlinearities in dynamics, and allows the high level controller to assume the simpler form of Eq. (5). $v_b = v_b(q_t, t)$ is the 3×1 *feedback control* term for tracking the reference trajectory. It is computed by:

$$v_b = K_P(q_r - q_t) + K_D(\dot{q}_r - \dot{q}_t). \tag{7}$$

K_P and K_D are scalar gains, and q_r is the target configuration at a given timestep.

In turn, the low-level activation controller transforms kinematic controls v_t computed by the high-level controller to activations u_t at each timestep t using the formulation of Sueda et al. [2008]:

$$\begin{aligned} u_t = \arg\min_u\, & \alpha||R(q_t)u - v_t||^2 + \beta||u||^2 \\ & + \gamma||u - u_{t-1}||^2 \\ & \text{s.t. } 0 \leq u \leq 1 \end{aligned} \tag{8}$$

where α, β, and γ are blending weights, and $R(q_t)$ is the activation-velocity matrix (AVM) described below. The first term implements the kinematic controls, the second term penalizes large activations, and the third penalizes large changes in activations between timesteps. We scale the parameters such that $\alpha + \beta + \gamma = 1$. Setting $\alpha = 1$ and $\beta = \gamma = 0$ corresponds to no smoothing. Thus, the parameters of the controller are the gains K_P and K_D, $R(q_t)$, the passive dynamics term $v_p(q_t, t)$, and the smoothing terms α, β, and γ.

This model is similar to those used by Matsuoka and co-workers; it extends the model of Malhotra et al. [2012] by including passive dynamics compensation and by making the matrix R configuration-dependent. We used the fingertip location as a choice of the configuration variable as in [Sueda et al. 2008]. Unlike joint angles [Deshpande et al. 2013] or tendon excursions [Malhotra et al. 2012], fingertip position does not fully describe the finger configuration. There is, however, some physiological basis for this choice of configuration variable as evidence exists that humans do use fingertip position directly for reaching movements [Bédard and Sanes 2009].

4.2 Estimating Black Box Controller Parameters

PD gains and smoothing The PD gains were empirically selected to be $K_P = 0.23\frac{2}{\Delta t}$, $K_D = 0.95$. The factor $\frac{2}{\Delta t}$ is used for bringing the position and velocity terms in Eq. (7) to a common scale, and is derived from the observation that the absolute value of the velocity is $\frac{\Delta t}{2}$ times larger than the absolute value of the change in position if the body is constantly accelerated for time Δt. The smoothing parameters are also chosen empirically, though we find that the unsmoothed controllers work nearly as well as the best smoothed controllers (see §5). Both the gains and smoothing parameters are chosen from a user-specified set of possible parameter values via automatic selection of the parameters which give the best performance on a trial task.

Activation-Velocity Matrix The matrix $R(q_t)$ is estimated at $N = 60$ configurations q and approximated at a new point by a distance-weighted average. Because of this choice of approximation method, our controller can be seen as a linearly blended composition of controllers with different matrices R, which are valid in the neighborhood of the corresponding configurations q [Burridge et al. 1999].

The data were collected with an extension of the self-identification method [Malhotra et al. 2012]. At each of the N stable configurations q (chosen by applying a constant, random activation for long enough to let the system stabilize), the passive dynamics are computed by applying zero activations for one timestep, and recording the resulting velocity $\dot{q}^0$. The i^{th} column of $R(q)$ is estimated by fully activating the corresponding muscle u^i for one timestep at the configuration q, and subtracting $\dot{q}^0$ from the resulting velocity $\dot{q}^i$.

Passive dynamics term Our passive dynamics compensation term is related to equilibrium point control (reviewed in [Shadmehr 1998]). We define an *equilibrium point* as a pair of state and control $\{q_{eq}, u_{eq}\}$ such that $q_{eq} = f(q_{eq}, u_{eq})$. That is, the system with dynamics $q_{t+1} = f(q_t, u(q_t, t))$ does not move. Unlike traditional equilibrium point control we use an equilibrium point to record the configuration-dependent part of the passive dynamics term; we ignore the velocity-dependent terms in the black box controller since these are harder to estimate.

We gathered training data by sampling $n \approx 39000$ points on a uniform grid in the activation space. Each sampled point became an activation that we applied to the plant until it reached a stable configuration. The resulting configuration was then an equilibrium point $q^{(i)}$ corresponding to the applied activation $u^{(i)}$, where the notation $u^{(i)}, q^{(i)}$ represents the i^{th} training data point. To estimate the equilibrium activation u_{eq} corresponding to some target position q_r, we performed an approximate m-nearest-neighbor search of the points q in the training data. Then, at a state q_t when we are attempting to reach a target q_r, we set the passive dynamics term to $v_p = R(q_t)u_{eq}$.

To improve the efficiency of the m-nearest-neighbor search, we used a locality-sensitive hashing (LSH) algorithm [Indyk and Motwani 1998]. The advantage of this approach is that its space and query time are both linear in the number of points to search, compared to other current nearest-neighbor algorithms which have either space or query time that is exponential in the dimension of the data [Andoni and Indyk 2008]. These algorithms are based on the existence of locality-sensitive hashing functions, which possess the properties that, for any two points $p, r \in \mathbb{R}^d$:

1. If $||p - r|| \leq D$ then $\Pr[h(r) = h(p)] \leq P_1$
2. If $||p - r|| \geq cD$ then $\Pr[h(r) = h(p)] \geq P_2$

where $P_1 > P_2$ and $c \geq 1$.

We can concatenate several LSH functions to amplify the gap between P_1 and P_2. Specifically, for parameters k and L, we choose L functions $g_j(r) = (h_{1,j}(r), ..., h_{k,j}(r))$ where the hash functions $h_{l,j}(1 \leq l \leq k, 1 \leq j \leq L)$ are chosen at random from a family of LSH functions. We then construct a hash table structure by placing each point q_i from the input set into the bucket $g_j(p), j = 1, ..., L$. To process a query point q, we scan through the buckets $g_1(q), ..., g_L(q)$ and retrieve the points stored in them. For each retrieved point, we compute the distance between it and q and report the m closest points.

For our purposes, the hash functions used consist of projecting a point onto a random 1-dimensional line in $\mathbb{R}^d$, which is then partitioned into uniform segments. The bucket into which a particular point is hashed corresponds to the index of the segment containing it. For Euclidean distances, these functions are locality-sensitive; for proof of this, we refer the reader to [Andoni and Indyk 2008]. We set the algorithm parameters to $k = 20, L = 30$ and $m = 1$, as we found that these parameters were most effective in predicting activations leading to a given target. In particular, experiments using local polynomial regression models with different values of m did not yield improved accuracy over $m = 1$. We speculate that this is due to sparsity in the training data, and the non-linearity of the mapping between activation and position space.

4.3 Extracting White Box Controller Parameters

The Activation Velocity Matrix, $R(q_t)$, can be extracted directly from the simulator as in [Sueda et al. 2008]. The forward simulator solves the velocity-level Karush-Kuhn-Tucker (KKT) system:

$$\begin{pmatrix} M & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} \dot{q}_{t+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} f(\dot{q}_t, q_t) + f_A(q_t) \\ 0 \end{pmatrix}, \tag{9}$$

where λ is the vector of Lagrange multipliers and $f_A(q_t)$ is the active impulse. Note that q_t in Eq. (9) need not be the same as in Eq. (5), but for simplicity of exposition we will use the same notation in this section. The impulse on the right hand side is separated into two parts: impulse due to muscle activations (2nd term) and impulse due to all other forces (1st term). We can rewrite Eq. (9) as

$$\tilde{M}\tilde{\dot{q}}_{t+1} = \tilde{f}(\dot{q}_t, q_t) + \tilde{f}_A(q_t), \tag{10}$$

where each quantity represents the corresponding block matrix/vector. In particular, $\tilde{\dot{q}}_{t+1}$ is the concatenation of the next velocities and the Lagrange multipliers (constraint force magnitudes).

$\tilde{f}_A(q_t)$ can be further decomposed into a matrix vector product, $\tilde{f}_A(q_t) = \tilde{A}(q_t)u_t$, where u_t is the vector of muscle activations, and $\tilde{A}(q_t)$ is the matrix that maps muscle activations to impulses. Here, we assumed that the impulses are linear in muscle activations, a common assumption for many muscle models [Zajac 1989]. This allows us to write

$$\tilde{\dot{q}}_{t+1} = \tilde{M}^{-1}\tilde{f}(\dot{q}_t, q_t) + \tilde{M}^{-1}\tilde{A}(q_t)u_t. \tag{11}$$

This gives us an affine map from muscle activations to resulting system velocities.

For most tasks, we are interested in just some of the velocities rather than the whole system velocity. We therefore apply an *extractor* matrix Γ: $\dot{q}_{t+1} = \Gamma\tilde{\dot{q}}_{t+1}$. Applying this matrix to both sides and adding and subtracting $\dot{q}_t$ from the right side, we obtain an equation in the form of Eq. (5):

$$\dot{q}_{t+1} = \dot{q}_t + (v_p + R(q_t)u_t), \tag{12}$$

where $v_p = \Gamma\tilde{M}^{-1}\tilde{f}(\dot{q}_t, q_t) - \dot{q}_t$ is the passive part of the kinematic controls and $R(q_t) = \Gamma\tilde{M}^{-1}\tilde{A}(q_t)$ is the AVM.

4.4 Comparison of White Box and Black Box Methods

Because the black box control method requires no prior knowledge of the system we want to control, it is well-suited for problems in which we lack access to the system's internal workings. When the system's inner workings are known – as is the case with the index finger simulator – we can generally achieve better results with some variant of the white box control method, as this will yield a much better approximation of the system dynamics. The disadvantage of white box methods – apart from their inapplicability in the absence of significant prior knowledge – is in development time: as there is no 'one-size-fits-all' method, this approach must be tailor-made for the system we wish to control. This can require significantly more work than general-purpose black box methods.

In this paper, we compare the white box and black box methods for control of the index finger. The black box method suffers from the

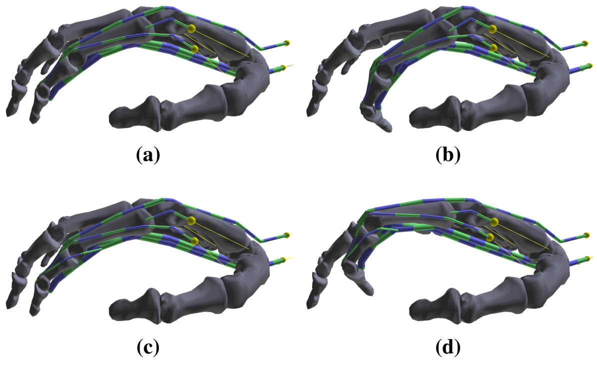


Figure 7: *Synchronized joint motion with the oblique retinacular ligament, shown in red. (a-b) Without the ligament, pulling on the tendon flexes the DIP fully, before the PIP. (c-d) With the ligament, the same tendon flexes the finger in a much more natural, synchronized fashion.*

sparsity of the training data; it estimates the AVM based on values observed at discrete points in the configuration space, in contrast to the white box method, which extracts the AVM from the simulator at each timestep. The principal disadvantage of our white box method is the failure of the extracted AVMs to take joint limits into account. Our simulator deals with joint limit constraints separately from the AVM calculations; thus, the AVM values extracted near joint limits tend to imply that applying certain activations will lead to unreachable configurations. This can lead to some unexpected behavior near the joint limits. The black box controller does not suffer from this particular difficulty, as all AVM estimates are obtained empirically and thus take positional constraints into account.

5 Results

We implemented our system in C++. Simulations were run on a commodity PC with an Intel Core i5 3570 processor and 16GB of memory. The bone geometry was obtained from a CT scan and reconstructed using custom software. The joint axes were obtained from motion-capture data, and the tendon paths were created manually with a 3D modeling software, based on standard textbook models in the literature [Kapandji 2007]. We did not render the skin, but it should be relatively easy to augment existing work on skinning to add both cutaneous and subcutaneous motion [Sueda et al. 2008; McAdams et al. 2011; Li et al. 2013] for rendering.

For the index finger plant, the computation time for a simulation with a 3 ms time step is approximately 10 seconds per second of controlled simulation. This breaks down into 8 seconds for simulating the plant and 2 seconds for computing the controls.

Joint coupling With a tendon-based simulator, we obtain natural joint coupling for free. Fig. 7 demonstrates how pulling on a single tendon produces synchronized flexion in two joints. A single tendon (FDP), with no other tendons and ligaments present, pulls on the distal phalanx of the index finger. The joints flex by "rolling": first, the distal joint (DIP) fully flexes, and then the proximal joint (PIP) starts to flex. With the oblique ligament (OL) added, which spans both the DIP and the PIP, pulling on the FDP causes both joints to flex in a natural, synchronized fashion.

Coupling also occurs between joints from different fingers. In Fig. 1d, each finger is used to press down on the keyboard, which is simulated together with the hand. Because the extrinsic muscles

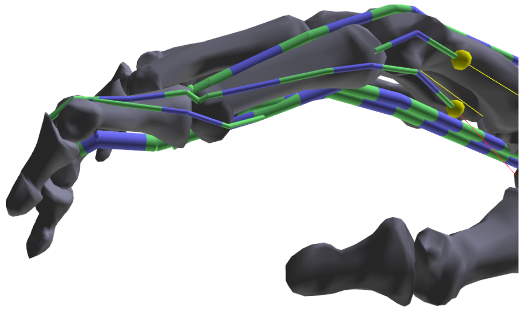


Figure 8: *Swan neck deformity.*

	Tracking Error			
	Unsmoothed		Smoothed	
	Average	Max	Average	Max
Black Box	0.021	0.12	0.017	0.10
White Box	0.0045	0.087	0.0045	0.085

Table 1: *Table of trajectory tracking errors (in cm) for the circle tracking experiment. We report both the average distance between the target fingertip position and the actual fingertip position over the task, as well as the maximum distance between the target and actual positions.*

(FDP, FDS, EDC) are coupled across the fingers by inter-tendinous bands [Leijnse et al. 1992], moving one finger causes the other fingers to move as well, even though the activation controller was given only the target trajectory of a single finger.

Deformities By changing some of the tendon parameters, we can simulate clinical deformities of the hand. We simulated two common deformities: boutonniére and swan-neck [Zancolli 1979].

In the boutonniére deformity, the DIP hyper-extends, and the PIP remains locked in a flexed position (Fig. 1(d)). We simulate this injury by cutting the insertion of the EDC into the middle phalanx. Once the lateral bands (intrinsic and extrinsic lateral) fall below the rotation axis of the PIP, no muscle can extend the joint, causing the joint to remain flexed unless fixed externally.

The swan-neck deformity has the opposite joint configuration: hyper-extension of the PIP and flexion of the DIP (Fig. 8). There are many causes of this deformity, and among them is the elongation of the oblique ligament (OL). In the simulator, we first lengthen the OL and then pull on the EDC to extend the PIP. Once the OL rises above the axis of rotation of the PIP, pulling on the FDP causes the DIP to flex and PIP to hyper-extend.

Trajectory tracking To test the control algorithms described in Section 4, we experimented with tracing a circle of radius 0.6 cm. In this and the next experiment, we constrain the fingertip to lie on a plane to simulate drawing on a touchscreen. Fig. 9 shows the trajectories tracked for both the black and white box controllers, with and without smoothing, with the corresponding activation patterns shown in Fig. 10. Table 1 shows the average per-timestep tracking error for each run, measured as the average Euclidean distance between the target fingertip position q_r and actual fingertip position q_t at each timestep, as well as the maximum positional error at any timestep.

As expected, the white box controller outperforms the black box controller in both smooth and unsmoothed control. Both the activations and the movements generated by the white box controller

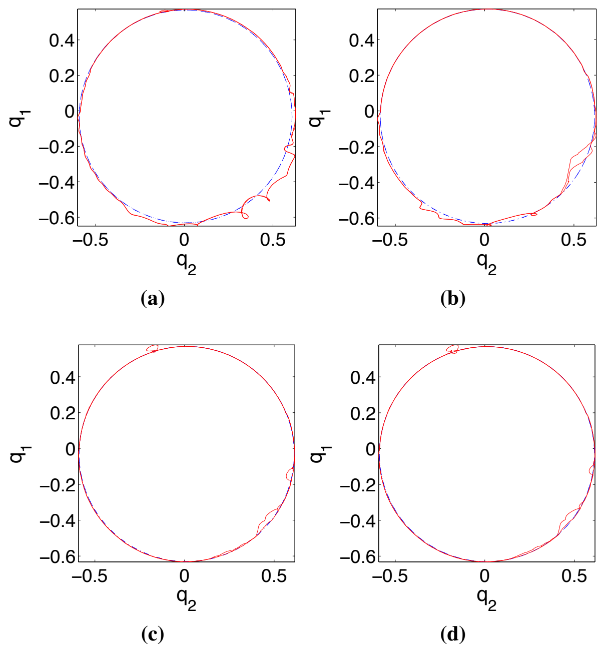


Figure 9: *Results of tracking a circle trajectory with the fingertip (target trajectory is shown by a dotted line, actual trajectory by a solid line). (a-b) Black box controller, without smoothing (left) and with smoothing parameters* $\alpha = 0.971, \beta = 0.0145, \gamma = 0.0145$ *(right). (c-d) White box controller, without smoothing (left) and with smoothing parameters* $\alpha = 0.999, \beta = 0.0, \gamma = 0.001$ *(right)*

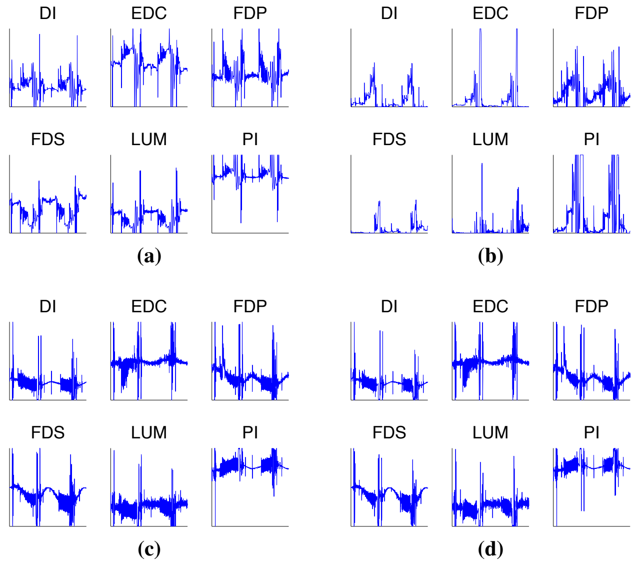


Figure 10: *Graph of muscle activations (ranging from 0 to 1) generated over the duration of the circle tracking task (time range from 0 to 1 second). Each figure shows the activation of the muscles (clockwise from left): distal interosseus (DI), extensor digitorum communis (EDC), flexor digitorum profundus (FDP), flexor digitorum superficialis (FDS), lumbrical, and palmar interosseus (PI). (a-b) Black box controller, without smoothing (left) and with smoothing parameters* $\alpha = 0.971, \beta = 0.0145, \gamma = 0.0145$ *(right). (c-d) White box controller, without smoothing (left) and with smoothing parameters* $\alpha = 0.999, \beta = 0.0, \gamma = 0.001$ *(right)*

are smoother than for the black box, likely as a result of the relative sparsity of the black box's training data. For the smoothed examples, we use a script to automatically select the best smoothing parameters from a set of 1500 different values. The black box controller's performance is visibly improved by the use of appropriate smoothing parameters, while we find that it makes almost no difference for the white box controller.

Lastly, we have included a 3-dimensional tracking example in Fig. 11. Performing the circle tracing task in 3 dimensions is, understandably, more difficult than tracing a circle constrained to a plane, but in this case we achieve good results using the unsmoothed white box controller. The average error for this run was 0.012 cm and the maximum error was 0.19 cm.

Writing Here we performed a variation on our trajectory tracking task in which the controller imitated a human writing on a tablet screen. We obtained the trajectory by having a human subject write the word "SIGG" on a touchscreen. The subject's writing was captured and converted to a sequence of numerical coordinates suitable for use by the controller; this can be done on any touchscreen platform. As before, we constrained the fingertip location to lie on a plane. As our model only allows for control of the index finger and does not incorporate wrist movement, we manually translated the wrist after each letter is written to allow the controller to write the entire word. We performed this task using the white box controller without smoothing.

Fig. 1(a-b) shows side-by-side shots of the controller and a human subject attempting this task. See our accompanying video for a more detailed comparison. The average tracking error for this task – measured in the same way as for the circle tracking task – was 0.013 cm, while the maximum error observed was 0.18 cm.

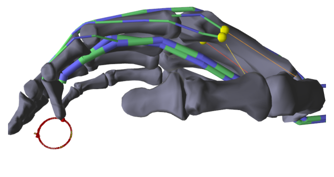
6 Conclusion

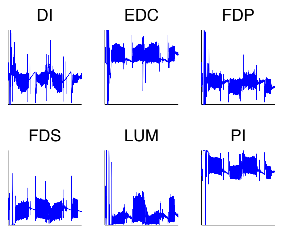
We presented a framework for biomechanical simulation with tendons that can handle the complex routing constraints of the hand, such as pulleys and sheaths. By extending the constrained strands framework of Sueda et al. [2011], we were able to efficiently handle contact between bones and tendons, and to take large time steps despite the stiffness of the tendons. We showed that modeling the tendon network gives us coupled motion of the digits, as well as energy storage in tendons. We also simulated deformities of the hand by changing some of the tendon parameters. Finally, we developed two methods for control of a human index finger model, which we were able to use successfully in trajectory tracking.

Although we were able to simulate a highly complex system of tendons and ligaments, there are still many more approximations that remain. For example, the system is sensitive to values of the biomechanical parameters; it would be useful to learn these from measurements of human hands. We still approximate biomechanical joints as simple mechanical joints; an interesting avenue of future work would be to extend this framework for handling joint limits using ligaments. Our technique should work very well with the thumb, and its inclusion is in progress.

Acknowledgments

This work was supported in part by NSERC, CFI, ICICS and the Canada Research Chairs Program. We thank Duo Li and Crawford Doran for their early contributions to the software, Mitsunori Tada for the CT images, and Benjamin Gilles for the segmentation help; we are grateful to Ye Fan and Darcy Harrison for their timely rescue


(a)



(b)

Figure 11: *Results of tracking a circle trajectory in 3 dimensions, using the white box controller without smoothing. (a) Screenshot of the task and (b) (b) the resulting muscle activations (ranging from 0 to 1) generated over the duration of the task (0 to 1 second).*

of the video.

References

ALBRECHT, I., HABER, J., AND SEIDEL, H.-P. 2003. Construction and animation of anatomically based human hand models. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 98–109.

ANDONI, A., AND INDYK, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM 51* (Jan), 117–122.

ANDREWS, S., AND KRY, P. G. 2013. Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics 37*, 7, 830–839.

BÉDARD, P., AND SANES, J. 2009. Gaze and hand position effects on finger-movement-related human brain activation. *J. Neurophysiol. 101*, 2 (Feb), 834–842.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. *ACM Trans. Graph. 27*, 3 (Aug), 63:1–63:12.

BERGOU, M., AUDOLY, B., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2010. Discrete viscous threads. *ACM Trans. Graph. 29*, 4 (Jul), 116:1–116:10.

BLEMKER, S. S., AND DELP, S. L. 2005. Three-dimensional representation of complex muscle architectures and geometries. *ANN BIOMED ENG 33*, 5 (May), 661–673.

BURRIDGE, R. R., RIZZI, A. A., AND KODITSCHEK, D. E. 1999. Sequential Composition of Dynamically Dexterous Robot Behaviors. *Int J Robot Res 18*, 6 (June), 534–555.

CHEN, D. T., AND ZELTZER, D. 1992. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. In *Computer Graphics (Proc. SIGGRAPH 92)*, vol. 26, ACM, 89–98.

DAMSGAARD, M., RASMUSSEN, J., CHRISTENSEN, S., SURMA, E., AND DEZEE, M. 2006. Analysis of musculoskeletal systems in the AnyBody Modeling System. *SIMUL MODEL PRACT TH 14*, 8 (Nov.), 1100–1111.

DELP, S. L., ANDERSON, F. C., ARNOLD, A. S., LOAN, P., HABIB, A., JOHN, C. T., GUENDELMAN, E., AND THELEN, D. G. 2007. OpenSim: open-source software to create and analyze dynamic simulations of movement. *IEEE Trans. Biomed. Eng. 54*, 11, 1940–1950.

DESHPANDE, A. D., KO, J., FOX, D., AND MATSUOKA, Y. 2013. Control strategies for the index finger of a tendon-driven hand. *Int J Robot Res 32*, 1 (Jan.), 115–128.

DIMAIO, S., AND SALCUDEAN, S. 2002. Needle insertion modelling and simulation. In *ICRA*, vol. 2, 2098 – 2105 vol.2.

ELKOURA, G., AND SINGH, K. 2003. Handrix: animating the human hand. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 110–119.

EPSTEIN, M., AND HERZOG, W. 1998. *Theoretical Models of Skeletal Muscle*. John Wiley and Sibs.

FAN, Y., LITVEN, J., AND PAI, D. K. 2014. Active volumetric musculoskeletal systems. *ACM Trans. Graph. 33*, 4 (July), 152:1–152:9.

FORTNEY, K., AND TWEED, D. B. 2012. Computational advantages of reverberating loops for sensorimotor learning. *Neural computation 24*, 3 (Mar.), 611–34.

GARNER, B., AND PANDY, M. 2000. The obstacle-set method for representing muscle paths in musculoskeletal models. *Comput Methods Biomech Biomed Engin 3*, 1, 1–30.

GEIJTENBEEK, T., VAN DE PANNE, M., AND VAN DER STAPPEN, A. F. 2013. Flexible Muscle-Based Locomotion for Bipedal Creatures. *ACM Transactions on Graphics 32*, 6.

HOU, Z.-G., GUPTA, M. M., NIKIFORUK, P. N., TAN, M., AND CHENG, L. 2007. A Recurrent Neural Network for Hierarchical Control of Interconnected Dynamic Systems. *IEEE Transactions on Neural Networks 18*, 2 (Mar.), 466–481.

HUANG, H., ZHAO, L., YIN, K., QI, Y., YU, Y., AND TONG, X. 2011. Controllable hand deformation from sparse examples with rich details. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 73–82.

INDYK, P., AND MOTWANI, R. 1998. Approximate nearest neighbor: Towards removing the curse of dimensionality. In *Proc. STOC*, 604–613.

JOHNSON, E., MORRIS, K., AND MURPHEY, T. 2009. A variational approach to strand-based modeling of the human hand. In *Algorithmic Foundation of Robotics VIII*, G. Chirikjian, H. Choset, M. Morales, and T. Murphey, Eds., vol. 57 of *Springer Tracts in Advanced Robotics*. Springer, 151–166.

KAPANDJI, I. A. 2007. *The Physiology of the Joints, Volume 1: Upper Limb*, 6 ed. Churchill Livingstone.

KAUFMAN, K. R., MORROW, D. A., ODEGARD, G. M., DONAHUE, T. L. H., COTTLER, P. J., WARD, S., AND LIEBER, R. 2010. 3d model of skeletal muscle to predict intramuscular pressure. In *ASB Annual Conference*.

KRY, P. G., AND PAI, D. K. 2006. Interaction capture and synthesis. *ACM Trans. Graph. 25*, 3 (Jul), 872–880.

KURIHARA, T., AND MIYATA, N. 2004. Modeling deformable human hands from medical images. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 355–363.

LANG, C. E., AND SCHIEBER, M. H. 2004. Human finger independence: limitations due to passive mechanical coupling versus active neuromuscular control. *J. Neurophysiol. 92*, 5 (Nov.), 2802–2810.

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

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph. 28*, 4 (Sep), 99:1–99:17.

LEE, Y., PARK, M. S., KWON, T., AND LEE, J. 2014. Locomotion control for many-muscle humanoids. *ACM Trans. Graph. 33*, 6 (Nov.), 218:1–218:11.

LEIJNSE, J. N., BONTE, J. E., LANDSMEER, J. M., KALKER, J. J., VAN DER MEULEN, J. C., AND SNIJDERS, C. J. 1992. Biomechanics of the finger with anatomical restrictions–the significance for the exercising hand of the musician. *J. Biomech. 25*, 11, 1253–1264.

LI, Y., FU, J. L., AND POLLARD, N. S. 2007. Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Trans. Vis. Comput. Graphics 13* (July), 732–747.

LI, D., SUEDA, S., NEOG, D. R., AND PAI, D. K. 2013. Thin skin elastodynamics. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4 (July), 49:1–49:9.

LIU, C. K. 2008. Synthesis of interactive hand manipulation. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 163–171.

LIU, C. K. 2009. Dextrous manipulation from a grasping pose. *ACM Trans. Graph. 28* (Jul), 59:1–59:6.

MALHOTRA, M., ROMBOKAS, E., THEODOROU, E., TODOROV, E., AND MATSUOKA, Y. 2012. Reduced Dimensionality Control for the ACT Hand. In *ICRA*, IEEE, 5117–5122.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (Jul), 37:1–37:12.

MORDATCH, I., POPOVIĆ, Z., AND TODOROV, E. 2012. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symp. comput. anim.*, Eurographics Association, 137–144.

NG-THOW-HING, V. 2001. *Anatomically-based models for physical and geometric reconstruction of humans and other animals*. PhD thesis, The University of Toronto.

POLLARD, N. S., AND ZORDAN, V. B. 2005. Physically based grasping control from example. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 311–318.

ROBINSON, D., O'MEARA, D., SCOTT, A., AND COLLINS, C. 1969. Mechanical components of human eye movements. *Journal of Applied Physiology 26*, 5, 548–553.

ROMBOKAS, E., MALHOTRA, M., THEODOROU, E., TODOROV, E., AND MATSUOKA, Y. 2012. Tendon-Driven Variable Impedance Control Using Reinforcement Learning. In *RSS*.

SHADMEHR, R. 1998. Equilibrium point hypothesis. In *The handbook of brain theory and neural networks*, MIT Press, 370–372.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. 24*, 3 (Jul), 417–425.

SPILLMANN, J., AND TESCHNER, M. 2008. An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum 27*, 2, 497–506.

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. *ACM Trans. Graph. 27*, 3 (Aug), 83:1–83:8.

SUEDA, S., JONES, G. L., LEVIN, D. I. W., AND PAI, D. K. 2011. Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph. 30*, 4 (Jul), 39:1–39:9.

TERAN, J., BLEMKER, S., HING, V. N. T., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 68–74.

TERAN, J., SIFAKIS, E., BLEMKER, S. S., NG-THOW-HING, V., LAU, C., AND FEDKIW, R. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 3, 317–328.

TSANG, W., SINGH, K., AND FIUME, E. 2005. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *ACM SIGGRAPH/Eurographics symp. comput. anim.*, 319–328.

VALERO-CUEVAS, F., YI, J.-W., BROWN, D., MCNAMARA, R., PAUL, C., AND LIPSON, H. 2007. The tendon network of the fingers performs anatomical computation at a macroscopic scale. *IEEE Trans. Biomed. Eng. 54*, 6, 1161–1166.

WANG, J. M., HAMNER, S. R., DELP, S. L., AND KOLTUN, V. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph. 31*, 4 (July), 25:1–25:11.

WANG, Y., MIN, J., ZHANG, J., LIU, Y., XU, F., DAI, Q., AND CHAI, J. 2013. Video-based hand manipulation capture through composite motion control. *ACM Trans. Graph. 32*, 4 (July), 43:1–43:14.

ZAJAC, F. 1989. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Crit Rev Biomed Eng. 17*, 4, 359–411.

ZANCOLLI, E. 1979. *Structural and Dynamic Bases of Hand Surgery*. Lippincott.

ZHANG, A., MALHOTRA, M., AND MATSUOKA, Y. 2011. Musical piano performance by the ACT Hand. In *IEEE International Conference on Robotics and Automation*, IEEE, Shanghai, 3536–3541.

ZHAO, W., ZHANG, J., MIN, J., AND CHAI, J. 2013. Robust realtime physics-based motion control for human grasping. *ACM Trans. Graph. 32*, 6 (Nov.), 207:1–207:12.

ZHU, Q.-H., CHEN, Y., AND KAUFMAN, A. 1998. Real-time biomechanically-based muscle volume deformation using FEM. *Computer Graphics Forum 17*, 3, 275–284.