Scalable Muscle-Actuated Human Simulation and Control

SEUNGHWAN LEE, Seoul National University
MOONSEOK PARK, Seoul National University Bundang Hospital
KYOUNGMIN LEE, Seoul National University Bundang Hospital
JEHEE LEE, Seoul National University

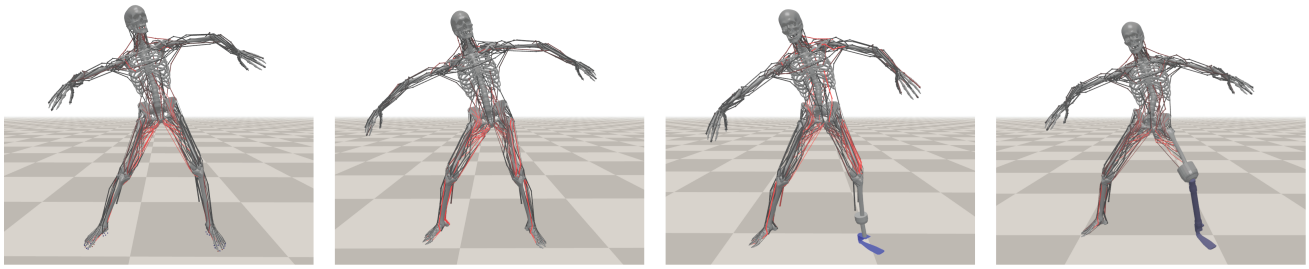

Fig. 1. Physics-based simulation and control of dynamic motor skills actuated by 284 to 346 musculotendon units. (Left to right) Musculoskeletal model with multi-segment feet, two-toe feet, and prosthetic legs.

Many anatomical factors, such as bone geometry and muscle condition, interact to affect human movements. This work aims to build a comprehensive musculoskeletal model and its control system that reproduces realistic human movements driven by muscle contraction dynamics. The variations in the anatomic model generate a spectrum of human movements ranging from typical to highly stylistic movements. To do so, we discuss scalable and reliable simulation of anatomical features, robust control of under-actuated dynamical systems based on deep reinforcement learning, and modeling of pose-dependent joint limits. The key technical contribution is a scalable, two-level imitation learning algorithm that can deal with a comprehensive full-body musculoskeletal model with 346 muscles. We demonstrate the predictive simulation of dynamic motor skills under anatomical conditions including bone deformity, muscle weakness, contracture, and the use of a prosthesis. We also simulate various pathological gaits and predictively visualize how orthopedic surgeries improve post-operative gaits.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Motion processing**.

Additional Key Words and Phrases: Anatomical Human Modeling, Musculoskeletal Modeling, Deep Reinforcement Learning, Joint Range of Motion Modeling, Locomotion Control, Gait Analysis

Authors' addresses: Seunghwan Lee, Department of Computer Science and Engineering, Seoul National University, Seoul, lsw9021@mrl.snu.ac.kr; Moonseok Park, Department of Orthopaedic Surgery, Seoul National University Bundang Hospital, Seoul, pmsmed@gmail.com; Kyoungmin Lee, Department of Orthopaedic Surgery, Seoul National University Bundang Hospital, Seoul, oasis100@snubh.org; Jehee Lee, Department of Computer Science and Engineering, Seoul National University, Seoul, jehee@mrl.snu.ac.kr.

0730-0301/2019/8-ART73
https://doi.org/

ACM Reference Format:
Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable Muscle-Actuated Human Simulation and Control. *ACM Trans. Graph.* 38, 4, Article 73 (August 2019), 13 pages.

1 INTRODUCTION

Human motion is affected by many anatomical factors such as the geometry of bones, muscle conditions, fatigue, habits, and even emotion. Small changes in anatomical conditions often alter the overall motion and result in distinctive movement patterns of each individual. The musculoskeleton of a human body is a highly-complex dynamic system. The human body has over 600 muscles and the half of them participate in joint movements. Muscle contraction and relaxation are a dynamic process of activating and deactivating tension-generating sites within muscle fibers. The brain sends excitation signal through the nervous system to activate and deactivate individual muscles and thus coordinates full-body movements.

This work aims to build a comprehensive musculoskeletal model and its control system that reproduces realistic human movements driven by muscle contraction dynamics. The variations in the model generate a wide spectrum of human movements ranging from normal (or typical) movements to highly stylistic variants, even to pathologic ones as well. The key technical challenges include accurate and comprehensive musculoskeletal modeling, the scalable and reliable simulation of anatomical features, and the robust control of the under-actuated dynamical system. Our model includes most of the skeletal muscles that serve for moving major joints. Our simulation system reliably deals with muscle contraction dynamics and joint range of motion (ROM) induced by background elasticity of muscles. We also present a new control algorithm based on Deep Reinforcement Learning (DRL).

Recently, DRL has shown its potentials for the control of physically-simulated articulated figures. The control policy represented by

deep neural networks has successfully reproduced highly-detailed human movements including walking, running, cartwheel, and flipping [Peng et al. 2018a]. It has been reported that the performance of DRL for continuous control depends on the choice of actuator types [Peng and van de Panne 2017]. It works better when the control policy outputs Proportional-Derivative (PD) targets rather than outputs joint torques directly. Policy learning becomes even more challenging if we are to learn a policy that generates activation levels of musculotendon actuators. There are successful cases of learning to run with a simple musculoskeletal model with merely 18 muscles [Kidziński et al. 2018]. However, its generalization to deal with a comprehensive 3D model has not been reported yet. In this work, we present a new two-level algorithm equipped with a hierarchical structure of policy networks. The skeleton layers learn the kinematics and dynamics of articulated skeletal motion at low frame rates, while the musculature layers learn muscle activations at higher frame rates. Our two-level algorithm is scalable to deal with the full-body musculoskeletal model with 346 muscles. We will demonstrate the effectiveness of our algorithm with various examples:

- Our simulation and control algorithm predicts how anatomical symptoms, such as bone deformity and contracture, affect full-body movements. Based on this capability, we will demonstrate stylistic, anatomic variations of human movements, including highly dynamic motor skills.
- We can predict and evaluate the effectiveness of prostheses by simulating their dynamic models with our musculoskeletal model. We experiment with both transtibial and transfemoral prostheses. Walking, running, and dancing with a prosthetic leg will be demonstrated.
- We also simulate and visualize various pathologic gaits and how orthopedic surgeries improve the gaits. Pre-operative gaits are constructed by providing anatomic conditions. We simulate muscle transplant and osteotomy (bone-editing) surgeries on our musculoskeletal model to predictively simulate post-operative gaits.

2 RELATED WORK

2.1 Musculoskeletal Modeling and Simulation

The simulation of musculoskeletal systems has been studied for several decades [Thelen et al. 2003; Zajac 1989]. Ever since prototype systems have been built and released [Damsgaard et al. 2006; Delp et al. 2007; Lloyd et al. 2012], muscle-based modeling and simulation have gained a lot of attention in both Biomechanics and Computer Graphics.

There have been a series of studies for modeling and simulation of specific body parts. Modeling the muscles at the neck and the head produces complex cervical activities [Lee and Terzopoulos 2006] and facial animation [Sifakis et al. 2005]. Modeling muscle fatigueness in lower body generates physiologically retargeted motions [Komura et al. 2000]. Pai and his colleagues explored the simulation of hand manipulation, including musculotendinous simulation with sliding constraints [Sueda et al. 2008] and Eulerian-on-Lagrangian simulation of constrained strands [Sachdeva et al. 2015].

Comprehensive biomechanical models of the human upper-body equipped with Hill-type actuators have been built and demonstrated [Lee et al. 2009]. Lee et al. [2018b] incorporated the simulation of elastic, volume-preserving materials into muscle modeling and demonstrated the control of dexterous manipulation skills actuated by volumetric muscles. Nakada et al. [2018] modeled human sensorimotor control using deep neural networks, which emulate neural pathways from visual perception to the activation of motor units.

There have been stream of research exploiting the kinematics and geometry of anatomical models. Saito et al. [2015] generated variations of human body shapes by simulating hypertrophy and atrophy of skeletal muscles. Multiple full-body 3D scans in various poses and a reference anatomical model can produce a personalized model [Kadleček et al. 2016]. A volumetric human body model learned from a collection of full-body 3D scans was able to produce realistic soft-tissue deformation in response to novel motions and external forces [Kim et al. 2017]. Akhter and Black [2015] collected a motion capture dataset that includes a variety of stretching poses and learned a pose-dependent model of joint limits. Jiang and Liu [2018] used the same dataset to learn a joint limit model represented by a fully-connected neural network.

2.2 Motion Control and DRL

Locomotion control of an under-actuated biped has been a long-standing research topic in Computer Graphics, Robotics, and Biomechanics. Rule-based [Yin et al. 2007], learning-based [Sok et al. 2007], data-driven [Lee et al. 2010], and optimization-based [Han et al. 2014] approaches have been explored to generate dynamically-stable, torque-driven locomotion imitating realistic human movements.

Muscle-driven locomotion control emerged with the help of stochastic optimization methods, such as CMA-ES (Covariant Matrix Adaptation Evolutionary Strategy). Wang et al. [2012] designed a lower-body model with eight musculotendon units in each leg, which generate torques only in the sagittal plane. They parameterized muscle control with hand-crafted feedback rules and optimized the control parameters using CMA-ES. Geijtenbeek et al. [2013] applied muscle-driven control to a variety of character morphologies. To do so, they optimized both the control parameters and muscle routing simultaneously to cope with varied morphologies. Lee et al. [2014] designed a comprehensive model with more than 100 musculotendon units. CMA-ES also played an important role of optimizing trajectories on top of low-level muscle control, for which Quadratic Programming (QP) effectively handled many actuated degrees of freedom. They also evaluated the robustness of the optimized controller against external pushes and found that the controller responds similarly to how humans respond to unexpected pushes. [Lee et al. 2015].

In 2007, the biped models had less than 10 actuated degrees of freedom (DOFs) [Sok et al. 2007; Yin et al. 2007]. The DOFs of the dynamic models increased more than tenfold by 2014 [Lee et al. 2014]. This trend still continues with the recent addition of deep network models [Nakada et al. 2018] and volumetric muscle modeling [Lee et al. 2018b]. The scalability of stochastic optimization

methods does not match the exponential growth of the complexity of dynamic models and thus Deep Reinforcement Learning (DRL) has gained great attention as its alternative.

A variety of DRL algorithms have demonstrated its promise in torque-actuated control problems, including biped locomotion [Peng et al. 2017; Yu et al. 2018], dynamic sports activities [Liu and Hodgins 2017, 2018; Peng et al. 2018a,b], exotic creatures [Won et al. 2017, 2018], and learning to dress [Clegg et al. 2018]. It has been shown that DRL algorithms can learn biped locomotion without any reference motion capture data, though the resulting motion may not look humanlike [Heess et al. 2017]. Given a reference motion, the learned control policy can reproduce high-quality human motion [Peng et al. 2018a,b]. Recently, generic neural network policies capable of learning diverse behaviors and multiple motor skills are attracting attention [Berseth et al. 2018; Merel et al. 2017; Wang et al. 2017].

Recent approaches exploiting hierarchical structures on DRL have shown their potential to improve the scalability of control systems. Levy et al. [2019] used multi-level hierarchies to accommodate sparse reward tasks. Vezhnevets et al. [2017] argued that decoupling end-to-end learning across multiple levels gains efficacy of learning by allowing it to utilize different resolutions of time. Bacon et al. [2017] studied option-based actor-critic models that are capable of learning both the internal policies and the termination conditions of options.

Despite great successes in torque-actuated control problems, DRL has been applied to muscle-actuated control problems in limited settings with a small number of musculotendon units [Driess et al. 2018; Kidziński et al. 2018; Peng and van de Panne 2017]. Even the state-of-the-art DRL algorithms do not scale well with the complexity of muscle-actuated control. Our learning algorithm reformulates QP-based low-level motor control and plugs it into the framework of DRL. We will demonstrate that incorporating this domain-specific control strategy into DRL significantly improves the performance and scalability of learning.

3 MUSCULOSKELETAL SYSTEM

Our musculoskeletal model includes a tree structure of rigid bones connected by 8 revolute joints (including knees and elbows) and 14 ball-and-socket joints (including hips, ankles, shoulders, and wrists). The model also includes 284 musculotendon units corresponding to skeletal muscles that contribute to joint motion in the skeleton (see Figure 2). Our model has no fingers and has two versions of foot models: *Two-toe* and *Multi-segment*. The articulation of the foot is simplified to have three segments for the two-toe foot, while the multi-segment foot features additional 12 revolute joints and 31 muscles for each foot. The muscles that do not contribute to any skeletal motion are omitted.

3.1 Muscle Model

The muscles are attached to bones on each end by tendons. The muscle attachment sites are called its *origin* (on the proximal side) and *insertion* (on the distal side). Muscle contraction pulls the bones on each side to move the joint inbetween them. The muscle geometry is often approximated as a polyline that starts at the origin of the muscle, passes through a sequence of *waypoints*, and ends at its insertion. The polyline better approximates the muscle length than the straight line between the origin and insertion (see Figure 3). The location of a waypoint is expressed by using the linear blending skinning (LBS) function.

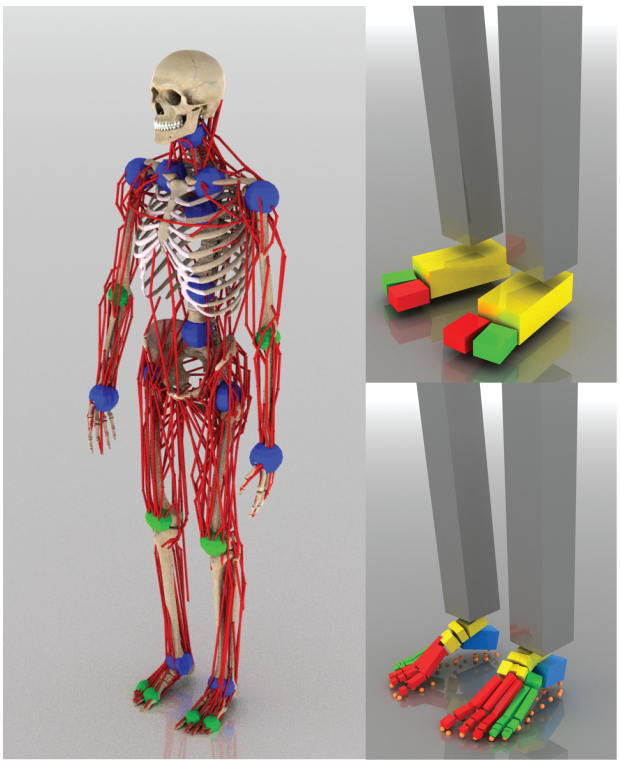

Fig. 2. Musculoskeletal Model. (Up right) Simplified two-toe foot. (Bottom right) Multi-segment foot.

$$\mathbf{p} = \sum w_j \mathbf{T}_j \mathbf{x}_j, \tag{1}$$

where $\mathbf{T}_j \in \mathbb{R}^{4\times4}$ is the transformation matrix of a bone computed through the kinematics of the skeleton, w_j is skinning weight, and $\mathbf{x}_j$ represents the coordinates relative to each bone coordinate system.

We employ a Hill-type muscle for the formulation of muscle contraction dynamics. According to the Hill-type model, contraction of muscle fibers generate tension f and its magnitude depends on muscle length l, the rate of its change $\dot{l}$, and the level of muscle activation $a \in [0, 1]$.

$$f = f(l, \dot{l}, a) = a \cdot f_l(l) \cdot f_v(\dot{l}) + f_p(l), \tag{2}$$

where $f_l(l)$ and $f_v(\dot{l})$ are force-length and force-velocity functions, respectively. The functions describe the maximum isometric tension of the muscle depending on its length and length changes. Even when the muscle is fully relaxed, the muscle develops passive force $f_p(l)$ because of its background elasticity. We refer the reader to the work by Thelen [2003] and Delp et al. [2007] for the details of force curves, muscle dynamics, and muscle parameters. We assume for simplicity non-stretchable tendons, which are very stiff bands of fibrous material and transfer muscle force to the attached bone. The

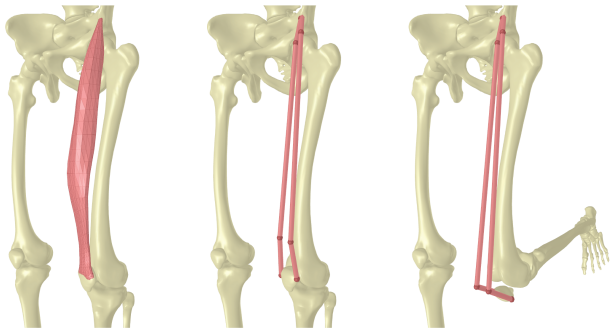

Fig. 3. Rectus Femoris muscle. (Left) 3D surface mesh. (Middle) Polyline approximation with waypoints. (Right) The LBS coordinates of the waypoints approximate the muscle route when the knee flexes.

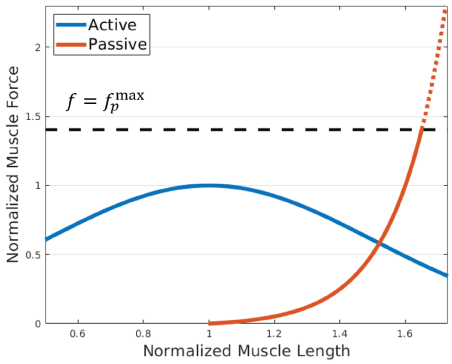

Fig. 4. Force-length curves in Hill-type muscles. (Blue) Maximum isometric force by active contraction of muscle fibers. (Red) Passive muscle-tendon tension when the fibers are inactive.

polyline transfers muscle force end-to-end through the waypoints. The waypoints work like pulleys to change the direction of force while conserving the momentum. We aggregate forces generated at the origin, insertion, and waypoints of muscle m into a single vector $\mathbf{f}_m(a)$.

3.2 Muscle-induced Joint limit

The range of joint motion is affected by many factors including bone/joint/ligament structures, geometric collision, and muscle tension. For example, the knee cap (a.k.a. Patella) prevents the knee from hyperextension, and the elasticity of a muscle would bound the motion of a joint as well. Muscle passive force $f_p(l)$ is negligible when the muscle length is below a certain threshold, but the force increases exponentially if the muscle stretches beyond the threshold (see Figure 4). Including such an exponentially-growing force in the physics-based simulation would result in a very stiff dynamic system that is prone to cause instability requiring very small simulation time step. Alternatively, we formulate muscle-induced joint limits by inequality constraints.

$$C_m(\mathbf{q}) := f_p^{\max} - f_p(l(\mathbf{q})) \geq 0 \tag{3}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the generalized coordinates of a full-body pose and $f_p^{\max}$ is a parameter that users specify. Similarly, we can also express other types of joint limits by inequality equations and aggregate all constraints into $\mathbf{C}(\mathbf{q})$. Explicitly enforcing the inequality constraints preemptively prevent the system from experiencing unstable conditions. The constraint velocity (the rate of change of the constraint) can be computed by:

$$\mathbf{v}_c := \frac{d\mathbf{C}}{dt} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} = \mathbf{J}_c \dot{\mathbf{q}} \tag{4}$$

We detail the derivation of $\mathbf{J}_c$ for our muscle-induced constraint in Appendix.

Given the constraint equation $\mathbf{C}$ and its velocity $\mathbf{v}_c$, the constraint force $\mathbf{f}_c$ that enforces the constraint can be computed by impulse-space simulation [Mirtich and Canny 1995], which leads to a Linear Complementary Problem (LCP). If constraints are active $C_i(\mathbf{q}) = 0$ or violated $C_i(\mathbf{q}) < 0$ for some i, solving for the complementary conditions computes the corresponding constraint forces:

$$\begin{aligned} \mathbf{v}_c &\geq 0 \\ \mathbf{f}_c &\geq 0 \\ \mathbf{v}_c^\top \mathbf{f}_c &= 0 \end{aligned} \tag{5}$$

The first condition guarantees that the constraints will not be violated any further in the next time step. The second condition guarantees that the constraint forces are not sticky or pulling. The last condition suggests that the constraint forces do not perform any work.

3.3 Musculoskeletal Dynamics

Given muscle and constraint forces, the Lagrangian dynamics of the musculoskeletal model can be described by

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_m \mathbf{J}_m^\top \mathbf{f}_m(a_m) + \mathbf{J}_c^\top \mathbf{f}_c + \tau_{\text{ext}} \tag{6}$$

where $\mathbf{q}$ is generalized coordinates, $\mathbf{M}(\mathbf{q})$ is the mass matrix, and $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ is Coriolis and gravitational forces. $\mathbf{f}_m$ and $\mathbf{f}_c$ are muscle and constraint forces, respectively. $\mathbf{J}_m$ and $\mathbf{J}_c$ are Jacobian matrices that map forces to generalized coordinates, and τ_{ext} is external force. Given muscle activation level $0 \leq a_m \leq 1$, Equation (6) solves for acceleration $\ddot{\mathbf{q}}$ to integrate pose $\mathbf{q}$ and its velocity $\dot{\mathbf{q}}$ over time.

Assuming that tendons are non-stretchable, muscle force consists of active contractile force linearly proportional to activation a_m and passive force independent of the activation, so $\mathbf{f}_m(a_m) = \frac{\partial \mathbf{f}_m}{\partial a_m} a_m + \mathbf{f}_m(0)$. We aggregate $\frac{\partial \mathbf{f}_m}{\partial a_m}$ and $\mathbf{f}_m(0)$ of all muscles, which are lumped into a matrix $\mathbf{A}$ and a vector $\mathbf{p}$ such that:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{c} = \mathbf{A}\mathbf{a} + \mathbf{p} + \mathbf{J}_c^\top \mathbf{f}_c + \tau_{\text{ext}} \tag{7}$$

where mth column of $\mathbf{A}$ is $\mathbf{J}_m^\top \frac{\partial \mathbf{f}_m}{\partial a_m}$, $\mathbf{p} = \sum_m \mathbf{J}_m^\top \mathbf{f}_m(0)$ and $\mathbf{a}$ is a vector that aggregates the activation levels of all muscles. Solving forward dynamics of the musculoskeletal model results in a linear mapping between $\ddot{\mathbf{q}}$ and $\mathbf{a}$.

$$\ddot{\mathbf{q}} = \mathbf{L}\mathbf{a} + \mathbf{b} \tag{8}$$

where $\mathbf{L} = \mathbf{M}^{-1}\mathbf{A}$ and $\mathbf{b} = \mathbf{M}^{-1}(\mathbf{p} + \mathbf{J}_c^\top \mathbf{f}_c + \tau_{\text{ext}} - \mathbf{c})$.

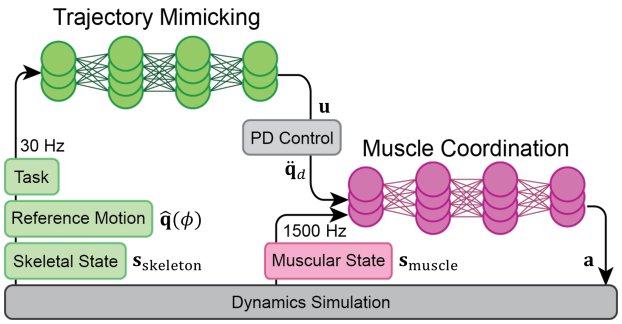

Fig. 5. Overview

4 CONTROL

To animate musculoskeletal models in physics-based simulation environments, we follow the trail of data-driven (a.k.a. example-guided) approaches [Lee et al. 2014; Peng et al. 2018a]. The user provides with a motion capture clip or a kinematic reference trajectory. Our goal is to learn a control policy (a.k.a. controller) that produces motions that resemble the reference data while achieving additional task objectives. The control policy $\pi(\mathbf{a}|\mathbf{s})$ coordinates activation levels of all muscles $\mathbf{a}$ at every time step in forward dynamics simulation to have the muscle-actuated character mimic the reference data. Here, state $\mathbf{s} = (\mathbf{s}_{\text{skeleton}}, \mathbf{s}_{\text{muscle}})$ includes the kinematic, dynamic, and anatomic states of the skeleton and musculotendon units.

In this section, we present a novel two-level architecture of policy learning that combines a state-of-the-art DRL method for trajectory mimicking with a novel supervised method for learning muscle coordination (see Figure 5). The trajectory mimicker is a stochastic policy $\pi_\theta(\mathbf{u}|\mathbf{s}_{\text{skeleton}})$ that produces PD target poses $\mathbf{u}$ as output given skeletal states $\mathbf{s}_{\text{skeleton}}$, where θ is network parameters to be optimized using DRL. PD servos $\ddot{\mathbf{q}}_d = PD(\mathbf{u})$ generate desired accelerations, which are passed to the second policy. The muscle coordinator $\mathbf{a} = \pi_\psi(\ddot{\mathbf{q}}_d, \mathbf{s}_{\text{muscle}})$ is a deterministic policy that activates muscles to generate desired motions, where ψ is network parameters determined by regression.

These two policies are jointly learned in a standard DRL framework with moderate overhead. Roughly speaking, learning a muscle-actuated control policy is three times as slow as learning a torque-actuated control policy due to the overhead of solving muscle contraction dynamics in simulation and evaluating the muscle coordination network. Note that PD targets serve as an intermediary between the two network policies in the learning process, but they are not used in actual simulations. Our simulation and control system at runtime is solely muscle-actuated requiring neither PD targets nor PD control at all. The trajectory mimicker learns and operates at the frame rate of the reference data, which is typically 30 frames per second. On the other hand, the muscle coordinator learns and operators at the rate of forward dynamics simulation, which is typically 900 to 1500 frames per second. Our learning algorithm interleaves two heterogeneous learning tasks to achieve end-to-end learning from anatomy-level control inputs all the way to full-body action and balance policies. Even though the trajectory mimicker takes only the skeletal state as input, the muscle states affect the skeletal state while generating transition tuples. Conversely, the muscle coordination policy also depends on the kinematics and dynamics of the skeletal model. In this way, the two policies collaboratively interact with each other to achieve maximum rewards in DRL.

4.1 Trajectory Mimicking

Trajectory mimicking can be formulated as a Markov decision process represented by a tuple $T = (S, U, P, R, \gamma, \rho_0)$, where $\mathbf{s} \in S$ is the kinematic and dynamic states of the skeleton, which explores an environment defined by transition probabilities $P : S \times U \mapsto S$, $\mathbf{u} \in U$ is an action that the agent takes, $r \in R$ is a reward, and ρ_0 is an initial state distribution. Note that we denote $\mathbf{s}_{\text{skeleton}}$ by $\mathbf{s}$ for simplicity only in this subsection. The agent explores the environment according to its policy $\pi_\theta(\mathbf{s})$ represented by parameters θ, and the environment evaluates the action with the reward r. The goal of the agent is to maximize expected cumulative rewards:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\mathbf{s}_0, \mathbf{u}_0, \mathbf{s}_1, \cdots}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t)\right] \tag{9}$$

where $\mathbf{s}_0 \sim \rho_0$, $\mathbf{u}_t \sim \pi_\theta(\mathbf{s}_t)$, and $\mathbf{s}_{t+1} \sim p(\mathbf{s}_t, \mathbf{u}_t)$. Since the evaluation of the objective requires intractable computational costs, typical remediation in DRL uses another neural network called value function $V^\pi(\mathbf{s})$ to approximate the accumulated rewards. The two networks $\pi_\theta(\mathbf{s})$ and $V^\pi(\mathbf{s})$ are iteratively updated at the training step [Lillicrap et al. 2015; Schulman et al. 2015, 2017]. We largely follows the implementation by Peng et al. [2018a] in defining states and rewards except for the change in the reward function that improves the output motion qualities.

The state is defined by $\mathbf{s} = (\mathbf{p}, \mathbf{v}, \phi)$, where $\mathbf{p}$ and $\mathbf{v}$ are the aggregations of the 3D position and linear velocity of bones, and $\phi \in [0, 1]$ is a phase variable which represents the normalized time elapsed in the reference motion. The position and linear velocity are represented in the moving coordinate system attached to the root body. The state vector is 112-dimensional. We specify actions by PD targets, which directly map to desired accelerations at joints.

$$\ddot{\mathbf{q}}_{\text{d}}(\mathbf{u}) = k_{\text{p}}(\mathbf{u} - \mathbf{q}) - k_{\text{v}}\dot{\mathbf{q}} \tag{10}$$

where k_{p} and k_{v} are gains of PD control.

The reward encourages the character to imitate the reference motion and optionally achieve task objectives simultaneously. The task objectives depend on the choice of the reference motion. The reward is defined by

$$r = w_q r_q r_e + w_g r_g \tag{11}$$

where r_q, r_e and r_g represent the pose imitation, end-effector imitation and task objectives, and w_q and w_g are their respective weights. The imitation rewards match the current and reference motions in terms of joint angles and end-effector positions. Let $\mathbf{q}$ be the generalized coordinates of the skeleton, and $\mathbf{p}_e$ be the position of end-effectors, that include LeftHand, RightHand, LeftFoot, RightFoot, and Head, relative to the moving coordinate frame attached

to the root.

$$r_q = \exp\big(-\sigma_q \sum_j \|\hat{\mathbf{q}}_j(\phi) \ominus \mathbf{q}_j\|^2\big)$$
$$r_e = \exp\big(-\sigma_e \sum_e \|\hat{\mathbf{p}}_e(\phi) - \mathbf{p}_e\|^2\big) \tag{12}$$

Here, the hat symbols indicates desired values taken from the reference data, j is the index of joints, and e is the index of end-effectors. The joint configurations are represented by unit quaternions and the quaternion difference is denoted by $\mathbf{q}_1 \ominus \mathbf{q}_2 = \ln(\mathbf{q}_2^{-1}\mathbf{q}_1)$ [Lee 2008]. In our experiments, the weights are $\sigma_q = 2.0$, $\sigma_e = 40.0$, $w_q = 0.9$ and $w_g = 0.1$.

Note that we multiply two imitation rewards r_q and r_e, while the task reward is added to them. The multiplication gets rewards when both terms are rewarded. It makes sense because joint angles and end-effector positions are tightly related with each other. The end-effectors of two motions are supposed to match well if their joint angles match well, and vice versa. High joint angle match reward r_q necessarily leads to high end-effector match reward r_e and thus they tend to reinforce each other. The relation between the imitation and task objectives is conflicting rather than reinforcing. The weighted sum of rewards allows each individual objectives to be pursued while searching for the compromise between conflicting goals.

4.2 Muscle Coordination

Muscle coordination is a problem of deciding activation levels for all muscles that achieve desired joint accelerations or torques. Since the human body has more muscles than minimally required to actuate joints, muscle coordination is an under-specified problem and thus infinitely many solutions exist. A standard solution method formulates the problem as Quadratic Programming (QP) that minimizes two objectives.

$$\begin{aligned} \min_{\mathbf{a}} \quad & \|\ddot{\mathbf{q}}_{\mathrm{d}}(\mathbf{u}) - \ddot{\mathbf{q}}(\mathbf{a})\|^2 + w_{\mathrm{reg}}\|\mathbf{a}\|^2 \\ \text{subject to} \quad & \mathbf{M}\ddot{\mathbf{q}} + \mathbf{c} = \sum_m \mathbf{J}_m^\top \mathbf{f}_m(a_m) + \mathbf{J}_{\mathrm{c}}^\top \mathbf{f}_{\mathrm{c}} + \tau_{\mathrm{ext}} \\ & 0 \le a_m \le 1 \quad \text{for} \quad \forall m. \end{aligned} \tag{13}$$

The first term encourages that a coordination of muscle activations generates desired accelerations and the second term regularizes large activations. The equality constraint ensures the equation of motion of the musculoskeletal model, while the inequality constraints enforce the normalized range $[0, 1]$ of muscle activations.

We do not solve for the QP explicitly, but reformulate the problem into the regression-by-supervised-learning framework, which can be incorporated into DRL. Let $\mathbf{a} = \pi_\psi(\ddot{\mathbf{q}}_{\mathrm{d}}(\mathbf{u}), \mathbf{s}_{\mathrm{muscle}})$ be a network policy that maps desired accelerations to muscle activations. Here, muscle state $\mathbf{s}_{\mathrm{muscle}} = (\mathrm{vec}(\mathbf{A}), \mathbf{p})$ encodes information as to converting muscle activations into muscle forces in the joint space such that $\sum_m J_m^\top \mathbf{f}_m(a_m) = \mathbf{A}\mathbf{a} + \mathbf{p}$. The matrix $\mathbf{A}$ should be vectorized to feed into the network policy. Since $\mathbf{A}$ is sparse and its structure is fixed, we compactly pack non-zero values in the vector conversion. Acceleration $\ddot{\mathbf{q}}$ is a function of $\mathbf{a}$ as shown in Equation (8). Incorporating Equation (8) into Equation (13), the loss function for training π_ψ is:

$$\mathrm{Loss}(\mathbf{a}(\psi)) = \mathbb{E}\big[\|\ddot{\mathbf{q}}_{\mathrm{d}}(\mathbf{u}) - \mathbf{L}\mathbf{a}(\psi) - \mathbf{b}\|^2 + w_{\mathrm{reg}}\|\mathbf{a}(\psi)\|^2\big] \tag{14}$$

Instead of modeling inquality constraints explicitly, we use a bounded activation function, such as sigmoid, at the output layer to enforce the normalized range of muscle activations. To solve for a regression network minimizing the loss function, we need to sample a large collection of tuples $(\mathbf{L}, \mathbf{b}, \mathbf{u}, \mathrm{vec}(\mathbf{A}), \mathbf{p})$. Since DRL for trajectory mimicking readily generates numerous episodes during training, we sample tuples for regression from the episodes. During learning, our algorithm alternates between the trajectory mimicker and the muscle coordinator to collect tuples and jointly update the policy and value networks using a stochastic gradient descent method.

Our formulation might look somewhat different from the standard formulation of regression or supervised learning, which is supposed to take ground truth values as input and minimizes the discrepancy between the ground truth and the network output. Since we want to compute activation levels as the output of the regression network, the ground truth activation $\mathbf{a}^*$ is required in the standard form. Even though there is no ground truth value in our formulation, our loss function can evaluate how good the network output is with respect to the objective of muscle coordination. The objective is provided in terms of PD target $\mathbf{u}$, the geometric alignments and physiological parameters of musculotendon units ($\mathbf{A}$ and $\mathbf{p}$), and the equation of motion ($\mathbf{L}$ and $\mathbf{b}$). The loss function alone without ground truth values is sufficient for regression.

The ability to remove inequality constraints from the QP formulation is an unexpected benefit of using neural networks. The most time-consuming step in solving QP is dealing with inequality constraints. Without the constraints, QP can be reduced to a system of linear equations that can be solved very efficiently. The use of a bounded network activation function allows us to reformulate the problem without constraints and thus simplifies the solution method. The standard network update and backpropagation steps can readily handle the unconstrained problem.

5 EXPERIMENTS AND RESULTS

5.1 Simulation Settings and Muscle Parameters

Our dynamic simulation is written in C++. Open source library DART is used to simulate skeletal dynamics [Lee et al. 2018a]. The implementation code and the data are available at https://github.com/lsw9021/MASS. Each bone is approximated using an oriented bounding box to estimate its inertia tensor and detect bone-to-bone and bone-to-ground collision. Our character is 170cm tall and weighs 72kg.

Our model has two versions of foot models. In our examples, we mostly used the simplified two-toe foot for its computational efficiency and the multi-segment foot was used only when we had to simulate delicate foot motion at extra computational costs. The multi-segment foot consists of two passive joints (Calcaneocuboid and Naviculocuneiform) and ten active joints between Metatarsal and Phalangeal bones. The sole is represented by linear blend skinning with respect to foot bones. The points on the sole collide and contact with the ground surface to support the body. The collision/contact response is computed by an LCP solver. To reduce

Table 1. Simulation parameters

Motion	Control Hz	Simulation Hz	Length(s)
Walk	30	900	1.05
Run	60	1500	0.55
Jump	30	1200	2.21
Dance	30	900	3.17
Deadlift	30	900	2.16
Cartwheel	60	1500	2.38
Kick	30	1500	1.92

unnecessary movements of individual toe joints, we sorted the toes into two groups and controlled each group as if they are a reduced deformable model [Liu and Hodgins 2018]. The passive joints are a spring-damper system that absorbs foot-ground impact. The simulation time step is 900Hz for most of the examples except for highly dynamic motor skills, such as Kick and Cartwheel, which requires smaller time step 1500Hz. The use of the multi-segment foot also requires small time steps (1200Hz) for simulation stability (see Table 1).

We constructed our musculoskeletal model starting from a human skeleton geometry. We annotated origins, insertions, and waypoints of all muscles and tuned their physical parameters including the maximum force, rest length, and muscle-tendon ratio of all muscles. The muscle is divided into multiple musculotendon units if it origins or inserts at multiple points or areas. A short, curved, thick muscle such as Deltoid and Glueus Maximus is also divided into 3 to 5 musculotendon units. The weights w_j of linear blend skinning were initially set to be inversely proportional to the distance to nearby joints and then tuned manually to avoid penetration into bones. We referred to the OpenSIM data [Arnold et al. 2010] to set the initial values of physical parameters and also tuned the values further to make the model viable for physics-based simulation.

5.2 Control System Settings

Our control system heavily relies on deep neural networks. We use an open source deep learning platform *Pytorch* to construct the networks [Paszke et al. 2017]. Our trajectory mimicking policy is trained using PPO [Schulman et al. 2017]. During training, Intel 4 core i7-7700 CPU generate tuples in parallel. Whenever 2048 tuples are collected, the neural networks are updated with a minibatch of size 128. We use NVIDIA 1070Ti GPU to accelerate the update and backpropagation of the neural networks. In our experiments, we stack four fully connected layers with 256 nodes for trajectory mimicking and five fully connected layers with 512 nodes for muscle coordination without dropout, and both are initialized using Xavier initialization. The policy and value networks are updated at learning rate 10^{-4}, which is linearly decreased to 0 when 20 million tuples are collected. The Gaussian noise with a diagonal convariance matrix is used for exploration during training and we set each standard deviation to be 10 percent of the full range of the motion. All the noises are ignored in run-time simulation.

The regression network for muscle coordination is updated jointly with the trajectory mimicking networks. We collect the same number of training tuples for trajectory mimicking and muscle coordination regardless of a discrepancy in their inference rates. We tested two types of activation functions for bounding the range of muscle activations to $[0, 1]$: Sigmoid function and hyperbolic tagent function followed by ReLU. We found that the latter works better in many cases. We suspect that the performance difference is related to initialization. The sigmoid function centers at 0.5, while the clipped hyperbolic tangent function is zero when its input is zero. The regression network is also updated at learning rate 10^{-4} and the minibatch size is 128. The learning takes 12 to 36 hours with 10 to 30 million tuples for challenging examples. We sometimes accelerate learning process by using the network parameters learned by a torque-actuated model as an initial guess.

5.3 Assorted Motor Skills

We learned assorted motor skills from reference motion capture data available on the web. The motion data clips were retargeted to our model using *Autodesk MotionBuilder*$^{\text{TM}}$. Table 1 shows the list of motion clips and simulation parameters.

Gait Cycle. Walking is one of the most fundamental movements of the human body. The gait cycle of human walking has been comprehensively studied and abundant biomechanical data acquired from human subjects are available. The eight phases of gait cycle are broadly accepted and the function of muscles at each phase is thoroughly analyzed [Zajac et al. 2003]. We compare our simulation results with the reference electromyography (EMG) data during gait cycle in Figure 6. Note that the EMG signal is a reliable source of measuring the activation and deactivation timing of muscles, but the magnitude of the signal is not accurate. So, we plotted the activation and deactivation of muscles in red lines. The plots show that our simulation results match the reference EMG data pretty well except for Tibialis Anterior, which is supposed to dorsiflex the ankle during swing phases to increase toe-ground clearance. High toe-ground clearance in dynamic walking tends to decrease tripping risk. The walking data in our experiments exhibit a relaxed gait with relatively low toe-ground clearance and therefore the ankle dorsiflexion is not accentuated.

Sargent Jump. We learned a vertical jump controller from a single reference motion clip that allows us to generate a continuous spectrum of vertical jump motions parameterized by target heights. To do so, the task defines a reward function:

$$r_g(\mathbf{s}) = e^{-\gamma(\hat{y}_{\text{COM}} - y_{\text{COM}}(\mathbf{s}))^2} + e^{-\gamma(\hat{y}_{\text{lf}} - y_{\text{lf}}(\mathbf{s}))^2} + e^{-\gamma(\hat{y}_{\text{rf}} - y_{\text{rf}}(\mathbf{s}))^2} \quad (15)$$

where $\hat{y}_{\text{COM}}$, $\hat{y}_{\text{lf}}$, and $\hat{y}_{\text{rf}}$ denote respective target heights for the center of mass and both feet, and shape parameter $\gamma = 40.0$ modulates task rewards. In the spirit of *curriculum learning*, we first learned the jump task captured in the reference data with $\hat{y}_{\text{COM}} = 1.3$ meter and $\hat{y}_{\text{lf}} = \hat{y}_{\text{rf}} = 1.0$ meter, and gradually increased the parameters by 0.01 to address incrementally more challenging tasks. The reference motion is also timewarped to match the increased target heights. The learning at each level increases target heights if the character hits the target and successfully lands in balance, or the learning terminates if there is no improvement in target height over one

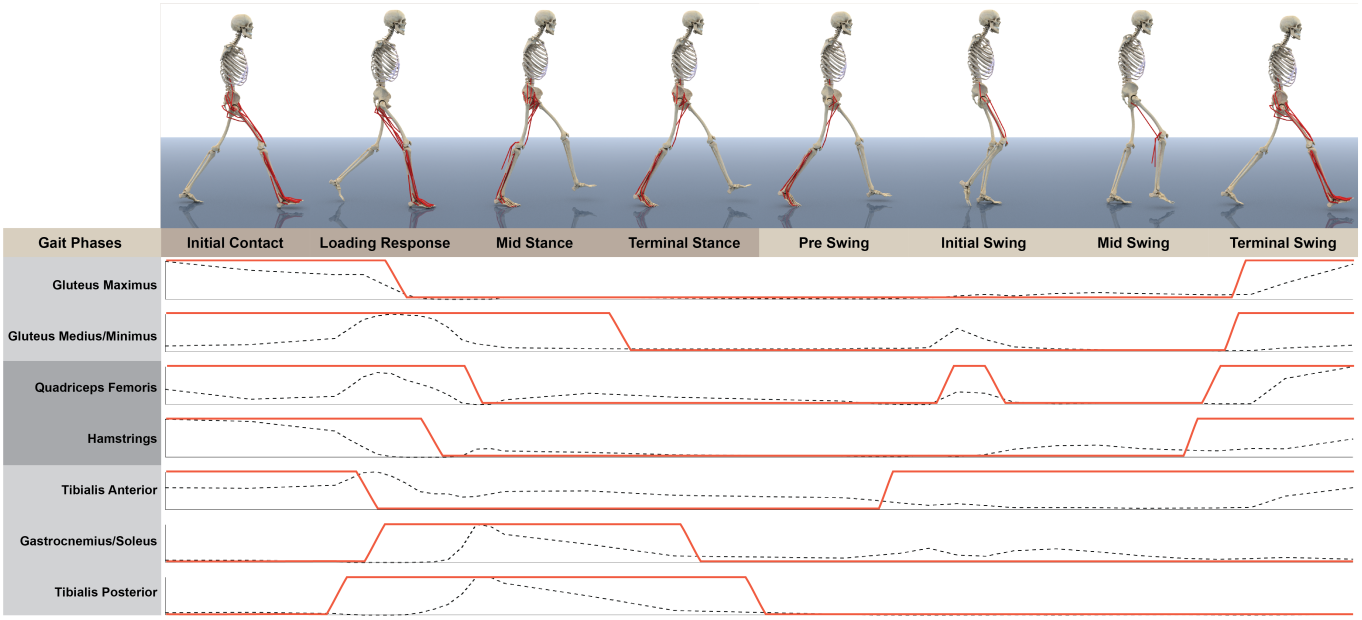

Fig. 6. The eight phases of gait cycle. Activation levels at lower-limb muscles are plotted in black dotted lines. The EMG reference data are shown in red solid lines.

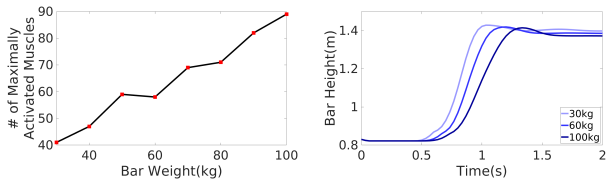

Fig. 7. The *Deadlift* example. (Left) The number of maximally activated muscles. (Right) The height of the bar relative to simulation time.

million tuples. Interestingly, the character learned to use arms more dynamically to jump higher.

Deadlift. The character learned to lift the bar with weights from the ground to the pelvis height. Similarly to the previous jump example, we gradually increased the weights to learn incrementally more challenging tasks. The bar is attached to the hands using zero-DOF joints. The simulated character has to use its full potential of muscle capabilities. Whenever the mass is increased, it has more muscles that maximally activate during motion (see Figure 7). Those maximally-activated muscles lose control over joint motion and therefore the controller has to learn a different muscle coordination for the increased mass. As the mass increases, the strenuous control response tends to get jerky. Our reward function provides a stationary objective of maintaining the balance of the bar.

$$r_g(\mathbf{s}) = e^{-\gamma ||\mathbf{p}_{\text{left}} - \mathbf{p}_{\text{right}}||^2} \tag{16}$$

where $\mathbf{p}_{\text{left}}$ and $\mathbf{p}_{\text{right}}$ are two-dimensional vectors from the center of the bar to the left and right weights, projected to the character's sagittal plane. The reward function penalizes the bar tilting up and down or leaning back and forth.

Prosthesis. We also learned walking, running, and dancing with a prosthetic leg (see Figure 9). Prosthesis is an artificial device that replaces the missing body parts. The goal of designing such a device is restoring the functionality of the missing body parts, capable of reproducing almost same movements of a healthy person. Starting from the motion of a normal person, our policy learning algorithm simulates the process of adapting to the prosthetic leg. We designed two types of lower extremity prostheses: Transtibial and transfemoral (see Figure 10). Both have a revolute joint with a passive spring damper system to model the compliant reaction of the prosthesis. The spring is stiff with coefficient 1000.0 and its damping coefficient $2\sqrt{1000.0}$ is set to critical damp.

5.4 Pathologic Gaits

Many pathologic gait patterns can be attributed to musculoskeletal conditions such as bone deformity and muscle deficiency. We consciously created such pathologic conditions in our model to see if gait patterns are generated as intended. Specifically, we implemented two types of conditions: Muscle contracture and femoral anteversion. *Contracture* is the shortening or stiffening of muscles, that results in decreased movements and range of motion. *Femoral anteversion* is an inward twisting of the femur (thigh bone), resulting in in-toeing gaits. Both symptoms can be easily incorporated into our model by adjusting the rest length of the muscle and twisting the geometry of the femur.

We want to simulate pathologic gaits starting from normal gait patterns. If the pathologic conditions are mild, our learning algorithm can adapt to the conditioned body automatically. However,

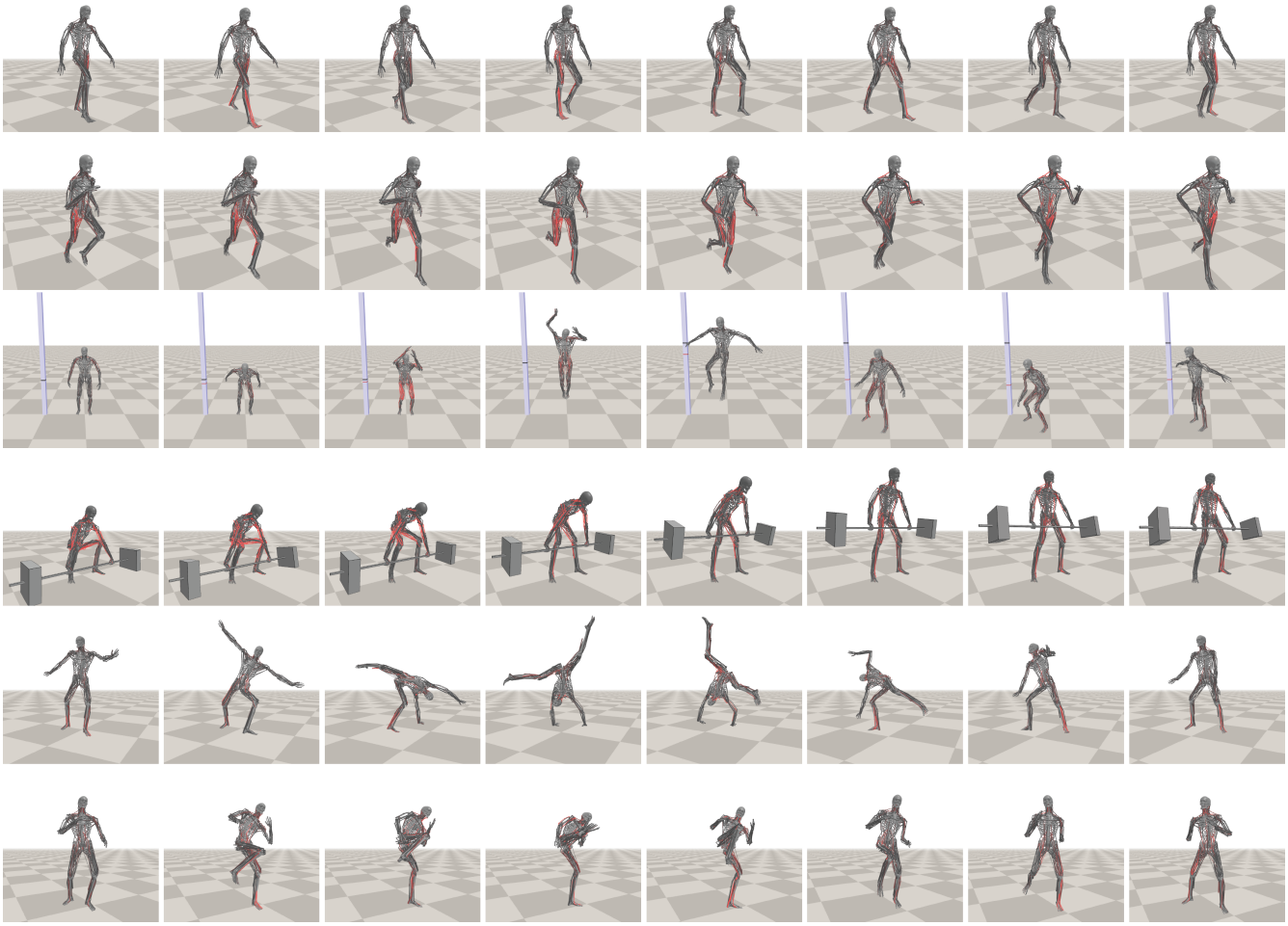

Fig. 8. Assorted motor skills: Walk, run, sargent jump, deadlift, cartwheel, and kick.

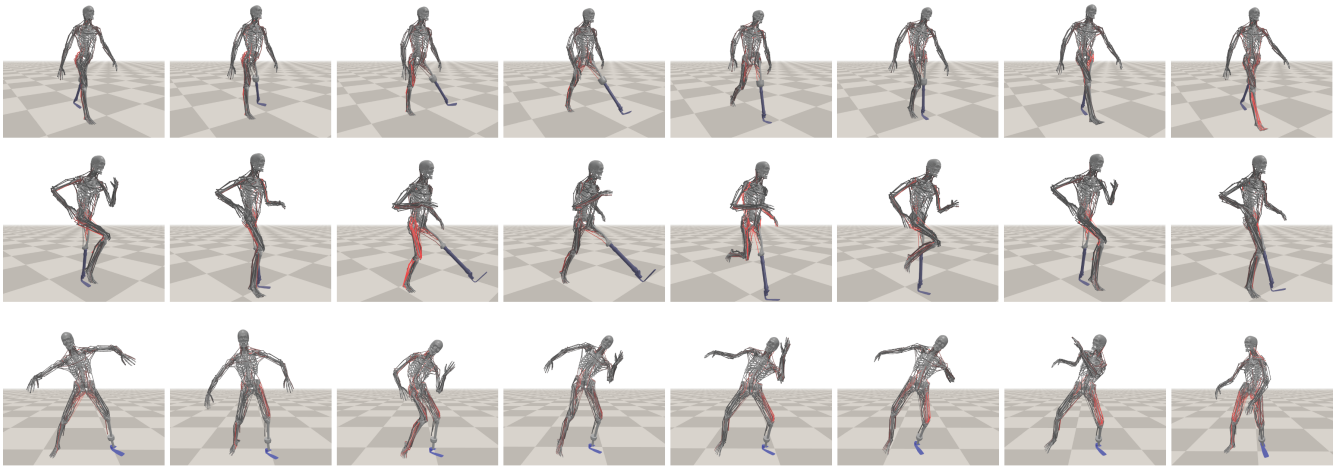

Fig. 9. Walking, running, and dancing with a prosthetic leg.

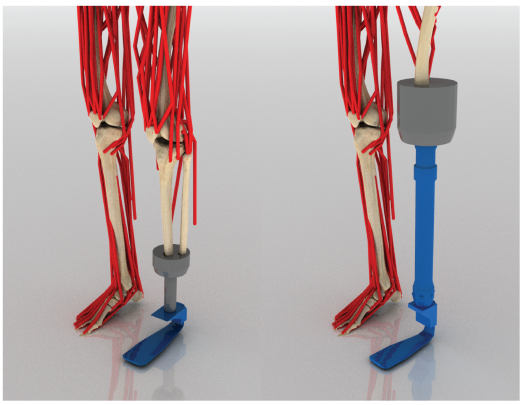

Fig. 10. Transtibial and transfemoral prostheses.

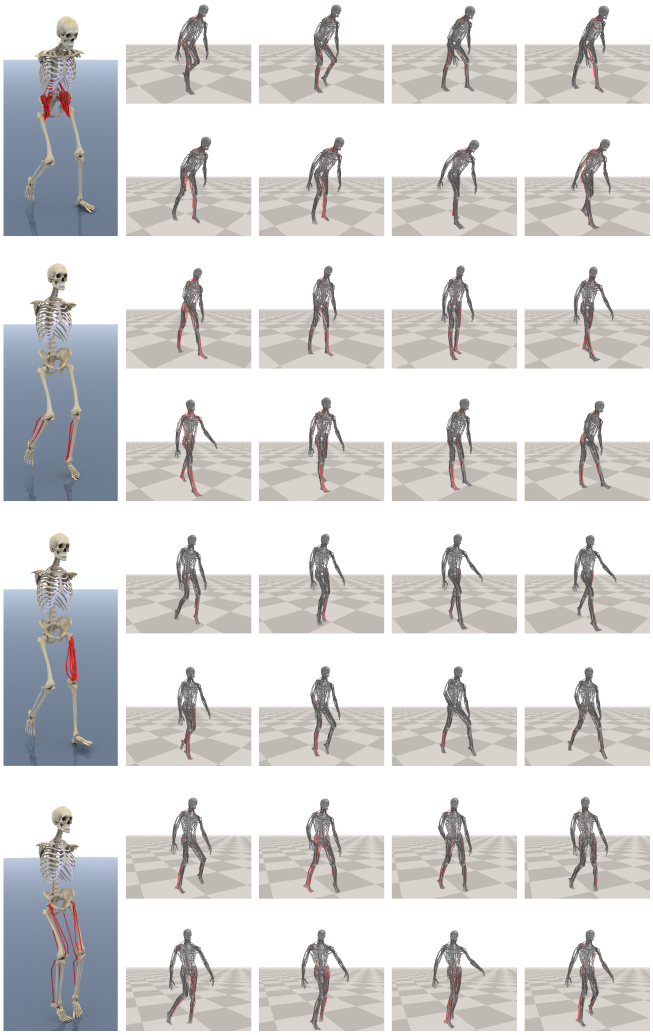

Fig. 11. Pathologic Gaits. (Top to Bottom) Hip flexion contracture, tip toe, asymmetric stiff knee, and multiple symptoms.

with severe conditions, the reference gait may violate the reduced range of motion at joints and the in-toeing stance foot may slide or penetrate through the ground surface. We can benefit from an optimization-based pre-processing phase that modifies the kinematic trajectory of the input gait to better fit to the conditions. The optimization is solved for every $\mathbf{q}_t$ of the input gait sequentially in a frame-by-frame manner.

$$\begin{aligned} \min_{\mathbf{q}_t} \quad & w_{\text{pos}}\|\mathbf{q}_t - \hat{\mathbf{q}}_t\|^2 + w_{\text{vel}}\|\mathbf{q}_t - \hat{\mathbf{q}}_{t-1}\|^2 + \\ & w_{\text{com}}\|\mathbf{p}_{\text{com}}(\mathbf{q}_t) - \frac{1}{2}(\mathbf{p}_{\text{LeftFoot}}(\hat{\mathbf{q}}_t) + \mathbf{p}_{\text{RightFoot}}(\hat{\mathbf{q}}_t))\|^2 \\ \text{subject to} \quad & C_m(\mathbf{q}_t) \geq 0 \quad \text{for} \quad \forall m, \end{aligned}$$

where $\hat{\mathbf{q}}_t$ denotes the reference motion. The first and second objectives penalize the deviations in position and velocity between the reference and output motions. The third objective encourages that the center of mass is nearby the support polygon. The inequality constraints enforce the reduced range of motion. We used IPOPT [Wachter and Biegler 2006] to solve for the frames.

With the aid of this optimization, we successfully learned four types of pathologic gaits (see Figure 11). The first example has the contracture of the psoas majors, which are strong hip flexors. Hence, their contracture results in permanent flexion of the hip joints. The second example has the contracture of the gastrocnemius and the soleus muscles, which results in stiff ankles and thus the character has to tip-toe. In the third example, the contracture of the major muscles in the left thigh results in stiff knees. Finally, the fourth example has a combination of multiple conditions, including the contracture of hamstring/quadriceps muscles in the thigh, the contracture of the triceps surae in the calf, and femoral anteversion. This combination of symptoms result in flexed knees, stiff ankles, and in-toeing feet.

Surgery Simulation. We simulated four types of orthopedic surgeries (TAL, RFT, DHL, and FDO), which are performed frequently to Cerebral Palsy patients (see Figure 12). Specifically, FDO (Femoral Derotational Osteotomy) is a procedure that corrects rotational deformities in the thigh and helps correct in-toeing and out-toeing during walking. The rectus femoris and hamstrings are large muscles in the front and back of the thigh, which have significant influence on walking. RFT (Rectus Femoris Transfer) and DHL (Distal Hamstring Lengthening) are procedures of transferring a muscle insertion to reduce the muscle tension. RFT and DHL together result in improving the range of motion in the knee joints. TAL (Tendo-Achilles Lengthening) is a procedure that lengthens the Achilles tendon to reduce the tension of the calf muscles, which can widen the range of motion of the ankle and consequently alleviate the symptoms of tiptoe walking. Implementing the effect of the surgeries is straightforward. We adjusted the torsion angle of the femur (FDO), moved the insertions of the rectus femories and semitendinosus (RFT and DHL), and changed the rest length of the Achilles tendon (TAL). The modified musculoskeletal model learned post-operative gaits that serve as predictive outcomes of the surgery simulation.

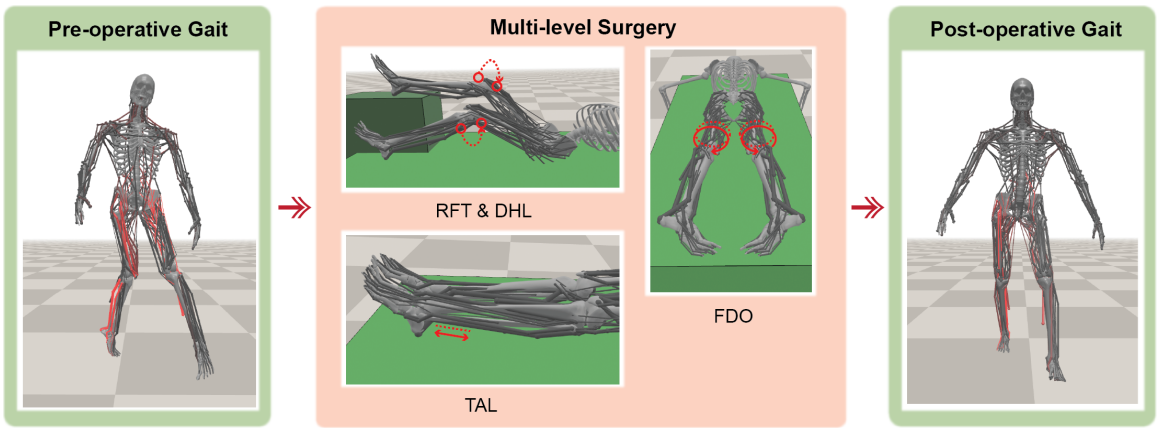

Fig. 12. Orthopedic surgery simulation.

6 DISCUSSION

Simulating virtual humans in physics-based simulation has been a long standing challenge in computer graphics. Our hierarchical network architecture enables reinforcement learning to address both long-term planning of trajectory mimicking and short-term muscle coordination in a unified framework, resulting in a scalable algorithm to simulate and control realistic human movements with highly-detailed musculoskeletal models.

Our algorithm scales remarkably well with the complexity of simulation models, though we have not tried to rigorously evaluate its asymptotic behavior. Each iteration of PPO includes 2048 frames of physics simulation accelerated by multi-threading. The torque-actuated model with 50 degrees of freedom and our musculoskeletal (two-toe foot) model with 284 muscles take 5.74 and 13.93 seconds for the iteration, respectively. Our examples requires 10 to 35 millions of tuples to learn their controller, taking about 12 to 36 hours of computation time. A low energy motion such as walking requires between 10 to 20 millions of tuples, while high energy motions such as cartwheel and jump require more experience tuples. The cartwheel example samples 35 millions tuples, taking 36 hours until its learning curve plateaus on a single PC. The use of multi-segment feet with 346 muscles requires about 40% more computation. Since runtime simulation cannot exploit multi-threading, the muscle-actuated simulation runs slightly slower than real-time.

Our framework also has numerous limitations. Our method heavily relies on domain-specific knowledge on anatomic modeling and physics-based simulation. The scalable end-to-end training of a full-body muscle-actuated motor skill without any domain knowledge is still an open problem. The successful anatomical simulation requires precise modeling of anatomical structures, careful tuning of kinematic, dynamic, and physiological parameters of musculotendon units and their geometric alignments. Currently, we rely on manual parameter tuning and incremental design refinements. The design and construction of an anatomic model viable for physics-based simulation is a challenging problem. It might be possible to develop an automatic procedure or algorithm that evaluates the functionality of musculotendon units and refines its geometric and physiologic parameters in accordance with its functionality.

Our multi-segment foot model added subtle, yet important details to simulated movements. The foot anatomy includes 26 bones connected by 33 joints, and numerous muscles, ligaments, and soft-tissue structures contribute to both active and passive (impact absorbing) behaviors of the foot. Our foot model still lacks a lot of important anatomical features. We found that the implementation of some passive features is beyond the scope of muscles and bones. Designing an anatomically accurate foot model would be an important corner stone for achieving the high-quality simulation of realistic human behavior, which poses a subject for future research.

We can think of many applications that can exploit our new technology. Our surgery simulation example shows the potential of our approach from the medical viewpoint. Predictive gait simulation can be a useful tool for medical doctors who treat patients with gait disturbance and plan surgical procedures for them. Medical doctors often have to decide which surgical procedures would be appropriate to the patient among several combinations available to the patient. Predictive gait simulation allows us to predict the outcomes of each surgical option and visualize the results.

ACKNOWLEDGMENTS

This work was supported by *Samsung Research Funding Center* under Project Number SRFC-IT1801-01.

REFERENCES

Ijaz Akhter and Michael J. Black. 2015. Pose-conditioned joint angle limits for 3D human pose reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1446–1455.

Edith M Arnold, Samuel R Ward, Richard L Lieber, and Scott L Delp. 2010. A model of the lower limb for analysis of human movement. *Annals of biomedical engineering* 38, 2 (2010), 269–279.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Glen Berseth, Cheng Xie, Paul Cernek, and Michiel Van de Panne. 2018. Progressive reinforcement learning with distillation for multi-skilled motion control. *arXiv preprint arXiv:1802.04765* (2018).

Alexander Clegg, Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. 2018. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics* 37, 6 (2018).

Michael Damsgaard, John Rasmussen, Søren T. Christensen, Egidijus Surma, and Mark D. Zee. 2006. Analysis of musculoskeletal systems in the AnyBody Modeling System. *Simulation Modelling Practice and Theory* 14, 8 (2006), 1100 – 1111.

Scott L. Delp, Frank C. Anderson, Anderson S. Arnold, Peter Loan, Ayman Habib, T. John, Eran Guendelman, and Darryl G. Thelen. 2007. OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement. *IEEE Transactions on Biomedical Engineering* 54, 11 (2007), 1940–1950.

Danny Driess, Heiko Zimmermann, Simon Wolfen, Dan Suissa, Daniel Haeufle, Daniel Hennes, Marc Toussaint, and Syn Schmitt. 2018. Learning to Control Redundant Musculoskeletal Systems with Neural Networks and SQP: Exploiting Muscle Properties. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6461–6468.

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. 2013. Flexible Muscle-Based Locomotion for Bipedal Creatures. *ACM Transactions on Graphics* 32, 6 (2013).

Daseong Han, Junyong Noh, Xiaogang Jin, Joseph S. Shin, and Sung Y. Shin. 2014. On-line real-time physics-based predictive motion control with balance recovery. In *Computer Graphics Forum*, Vol. 33. 245–254.

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

Yifeng Jiang and C Karen Liu. 2018. Data-driven approach to simulating realistic human joint constraints. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1098–1103.

Petr Kadleček, Alexandru-Eugen Ichim, Tiantian Liu, Jaroslav Křivánek, and Ladislav Kavan. 2016. Reconstructing personalized anatomical models for physics-based body animation. *ACM Transactions on Graphics* 35, 6, Article 213 (2016).

Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael Ong, Zhewei Huang, et al. 2018. Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. *arXiv preprint arXiv:1804.00361* (2018).

Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J. Black, and Sung-Hee Lee. 2017. Data-driven Physics for Human Soft Tissue Animation. *ACM Transactions on Graphics* 36, 4, Article 54 (2017), 12 pages.

Taku Komura, Yoshihisa Shinagawa, and Tosiyasu L Kunii. 2000. Creating and retargetting motion by the musculoskeletal human body model. *The Visual Computer* 16, 5 (2000), 254–270.

Jehee Lee. 2008. Representing Rotations and Orientations in Geometric Computing. *IEEE Computer Graphics and Applications* 28, 2 (2008), 75–83.

Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. 2018a. DART: Dynamic Animation and Robotics Toolkit. *The Journal of Open Source Software* 3, 22 (2018), 500.

Sunghee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics* 28, 4, Article 99 (2009).

Seunghwan Lee, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. 2018b. Dexterous Manipulation and Control with Volumetric Muscles. *ACM Transactions on Graphics* 37, 4, Article 57 (2018), 13 pages.

Sung Hee Lee and Demetri Terzopoulos. 2006. Heads Up!: Biomechanical Modeling and Neuromuscular Control of the Neck. *ACM Transactions on Graphics* 25, 3 (2006), 1188–1198.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Transactions on Graphics* 29, 4 (2010), 129.

Yoonsang Lee, Kyungho Lee, Soon-Sun Kwon, Jiwon Jeong, Carol O'Sullivan, Moon Seok Park, and Jehee Lee. 2015. Push-Recovery Stability of Biped Locomotion. *ACM Transactions on Graphics* 34, 6 (2015).

Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics* 33, 6, Article 218 (2014).

Andrew Levy, Robert Platt, and Kate Saenko. 2019. Hierarchical reinforcement learning with hindsight. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

Libin Liu and Jessica Hodgins. 2017. Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning. *ACM Transactions on Graphics* 36, 3 (2017).

Libin Liu and Jessica Hodgins. 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Transactions on Graphics* 37, 4 (2018).

John E. Lloyd, Ian Stavness, and Sidney Fels. 2012. ARTISYNTH: a fast interactive biomechanical modeling toolkit combining multibody and finite element simulation.

Josh Merel, Yuval Tassa, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. 2017. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201* (2017).

Brian Mirtich and John Canny. 1995. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*. 181–ff.

Masaki Nakada, Tao Zhou, Honglin Chen, Tomer Weiss, and Demetri Terzopoulos. 2018. Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation. *ACM Transactions on Graphics* 37, 4, Article 56 (2018), 15 pages.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Transactions on Graphics* 37, 4, Article 143 (2018).

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Transaction on Graphics* 36, 4, Article 41 (2017).

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: Reinforcement Learning of Physical Skills from Videos. *ACM Transactions on Graphics* 37, 6, Article 178 (2018).

Xue Bin Peng and Michiel van de Panne. 2017. Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter?. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17)*. Article 12, 13 pages.

Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K Pai. 2015. Biomechanical simulation and control of hands and tendinous systems. *ACM Transactions on Graphics* 34, 4, Article 42 (2015).

Shunsuke Saito, Zi-Ye Zhou, and Ladislav Kavan. 2015. Computational bodybuilding: Anatomically-based modeling of human bodies. *ACM Transactions on Graphics* 34, 4, Article 41 (2015).

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Eftychios Sifakis, Igor Neverov, and Ronald Fedkiw. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Transactions on Graphics* 24, 3 (2005).

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics* 26, 3 (2007).

Shinjiro Sueda, Andrew Kaufman, and Dinesh K. Pai. 2008. Musculotendon Simulation for Hand Animation. *ACM Transactions on Graphics* 27, 3, Article 83 (2008), 8 pages.

Darryl G Thelen et al. 2003. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Transactions-American Society Of Mechanical Engineers Journal Of Biomechanical Engineering* 125, 1 (2003), 70–77.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 3540–3549.

Andreas Wachter and Lorenz T. Biegler. 2006. On the Implementation of an Interior-point Filter Line-search Algorithm for Large-scale Nonlinear Programming. *Math. Program.* 106, 1 (2006), 25–57.

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-based Actuators and Objectives. *ACM Transaction on Graphics* 31, 4, Article 25 (2012).

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. 2017. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*. 5320–5329.

Jungdam Won, Jongho Park, Kwanyu Kim, and Jehee Lee. 2017. How to Train Your Dragon: Example-guided Control of Flapping Flight. *ACM Transactions on Graphics* 36, 6, Article 198 (2017), 13 pages.

Jungdam Won, Jungnam Park, and Jehee Lee. 2018. Aerobatics Control of Flying Creatures via Self-regulated Learning. In *SIGGRAPH Asia 2018 Technical Papers (SIGGRAPH Asia '18)*. Article 181.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Transaction on Graphics* 26, 3, Article 105 (2007).

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-energy Locomotion. *ACM Transactions on Graphics* 37, 4, Article 144 (2018), 12 pages.

Felix E Zajac. 1989. Muscle and tendon Properties models scaling and application to biomechanics and motor. *Critical reviews in biomedical engineering* 17, 4 (1989), 359–411.

Felix E Zajac, Richard R Neptune, and Steven A Kautz. 2003. Biomechanics and muscle coordination of human walking: part II: lessons from dynamical simulations and clinical implications. *Gait & posture* 17, 1 (2003), 1–17.

APPENDIX: DERIVATION OF MUSCLE-INDUCED JOINT LIMIT

To embed our inequality constraint into a standard LCP solver, we need to compute Jacobian $\mathbf{J}_c^m$ that maps velocities in joint space to velocities in constraint manifold. In our case, the Jacobian can be computed through the chain rule using equation (3):

$$\mathbf{J}_c^m = \frac{\partial C_m}{\partial \mathbf{q}} = \frac{\partial C_m}{\partial l}\frac{\partial l}{\partial \mathbf{q}} = -\frac{\partial f_p}{\partial l}\frac{\partial l}{\partial \mathbf{q}} \tag{17}$$

Let l_0^{muscle} and l_0^{tendon} be the optimal length of muscle fibers and tendon slack length, respectively. In Equation (17), the first term $\frac{\partial f_p}{\partial l}$ is derived by analytic differentiation.

$$\begin{aligned}\frac{\partial f_p}{\partial l} = \frac{\partial f_p(\tilde{l})}{\partial \tilde{l}}\frac{\partial \tilde{l}}{\partial l} &= \frac{\partial}{\partial \tilde{l}}\left[\frac{e^{k_{\text{PE}}(\tilde{l}-1)/\epsilon_{\text{m}}^0}-1}{e^{k_{\text{PE}}}-1}\right]\frac{1}{l_0^{\text{muscle}}} \\ &= \frac{1}{l_0^{\text{muscle}}}\cdot\frac{k_{\text{PE}}}{\epsilon_{\text{m}}^0}\cdot\frac{e^{k_{\text{PE}}(\tilde{l}-1)/\epsilon_{\text{m}}^0}}{e^{k_{\text{PE}}}-1}\end{aligned} \tag{18}$$

where $\tilde{l} = (l - l_0^{\text{tendon}})/l_0^{\text{muscle}}$ is the normalized muscle length, $k_{\text{PE}} = 4.0$ and $\epsilon_{\text{m}}^0 = 0.6$ are shape parameters that modulate passive force curves. The second term $\frac{\partial l}{\partial \mathbf{q}}$ is derived by

$$\begin{aligned}\frac{\partial l}{\partial \mathbf{q}} &= \sum_i \frac{\partial}{\partial \mathbf{q}}\|\mathbf{p}_{i+1}-\mathbf{p}_i\| \\ &= \sum_i \frac{1}{2\|\mathbf{p}_{i+1}-\mathbf{p}_i\|}\frac{\partial}{\partial \mathbf{q}}\left[(\mathbf{p}_{i+1}-\mathbf{p}_i)^\top(\mathbf{p}_{i+1}-\mathbf{p}_i)\right] \\ &= \sum_i \left[\frac{\mathbf{p}_{i+1}-\mathbf{p}_i}{\|\mathbf{p}_{i+1}-\mathbf{p}_i\|}\right]^\top \frac{\partial(\mathbf{p}_{i+1}-\mathbf{p}_i)}{\partial \mathbf{q}} \\ &= \sum_i \left[\frac{\mathbf{p}_{i+1}-\mathbf{p}_i}{\|\mathbf{p}_{i+1}-\mathbf{p}_i\|}\right]^\top \left[\frac{\partial \mathbf{p}_{i+1}}{\partial \mathbf{q}} - \frac{\partial \mathbf{p}_i}{\partial \mathbf{q}}\right]\end{aligned} \tag{19}$$

where $\mathbf{p}_i$ is a waypoint. Each small Jacobian $\frac{\partial \mathbf{p}_i}{\partial \mathbf{q}}$ is the partial sum of body Jacobians $\frac{\partial \mathbf{T}_j}{\partial \mathbf{q}}$ with the weights of linear blend skinning.