공학박사학위논문

Reproducing Human Movements via Physics-based Simulation of Human Sensorimotor System

물리 기반 사람의 감각운동계 시뮬레이션을 통한 자연스러운 사람 움직임 재현

2022 년 8 월

서울대학교 대학원

컴퓨터 공학부

이 승 환

Reproducing Human Movements via Physics-based Simulation of Human Sensorimotor System

물리 기반 사람의 감각운동계 시뮬레이션을 통한 자연스러운 사람 움직임 재현

지도교수 서진욱

이 논문을 공학박사 학위논문으로 제출함

2022 년 7 월

서울대학교 대학원

전기.컴퓨터 공학부

이승환

이승환의 박사 학위논문을 인준함

2022 년 7 월

위 원 장	김건희	(인)
부위원장	서진욱	(인)
위 원	이영기	(인)
위 원	이성희	(인)
위 원	이제희	(인)

Abstract

Many factors in human sensorimotor system contribute to human movements. In this thesis, we simulate human musculoskeletal system and tactile sensorimotor system to model intrinsic mechanisms underlying in the system that contributes to reliable human movements. The variations in the model generate motions such as walking, running to highly stylistic variants, and even pathologic ones as well. We can also generate a wide spectrum of movements driven by sensorimotor control ranging from passive reactions to external physical perturbations, to active manipulations with clear intentions.

This thesis covers three topics. Since human musculoskeletal system is such a large-scale system where there are more than 200 bones and 600 muscles, we first discuss the scalability of the musculoskeletal simulation. Second, we discuss methodologies for efficient simulation and control of volumetric muscles to incorporate the true mechanics of the musculoskeletal system. Lastly, we are to design a dynamic process of the sensorimotor control driven by tactile information to reproduce various physically reliable movements when interactions. We demonstrate various examples that manifest our framework can be a useful tool for understanding, analyzing, and reproducing the functionality of human systems that contribute movements.

Keywords: Computer Animation, Physics Simulation, Physics-based Character Animation, Deep Reinforcement Learning, Muscle Model, Musculoskeletal Model, Force Control, Impedance Control

Student Number: 2016-27526

Contents

Abstract **i**

Contents **ii**

List of Figures **v**

List of Tables **xi**

Chapter 1 Introduction **1**

Chapter 2 Background **9**

2.1 Musculoskeletal Simulation 10

2.1.1 Simulation of nonlinear volumetric solids 11

2.2 Motion Control 12

2.2.1 Learning Motor Skills with DRL 13

2.2.2 Force Control 15

Chapter 3 Scalable Muscle-Actuated Human Simulation and Control **17**

3.1 Overview 18

3.2 Musculoskeletal System 20

3.2.1 Muscle Model 21
3.2.2 Muscle-induced Joint limit 22
3.2.3 Musculoskeletal Dynamics 25
3.3 Control 26
3.3.1 Trajectory Mimicking 28
3.3.2 Muscle Coordination 30
3.4 Experimental Results 32
3.4.1 Simulation Settings and Muscle Parameters 32
3.4.2 Control System Settings 33
3.4.3 Assorted Motor Skills 35
3.4.4 Pathologic Gaits 38
3.5 Discussion 43

Chapter 4 Dexterous Manipulation and Control with Volumetric Muscles 46

4.1 Overview 47
4.2 Human model 49
4.2.1 FEM simulation 50
4.2.2 Muscle Model 52
4.2.3 Skeleton model 57
4.2.4 Coupling 58
4.3 control 58
4.3.1 Low-level controller 59
4.3.2 Jacobian Computation 62
4.3.3 High-level Controller 63
4.4 Experimental results 64
4.4.1 Muscle Geometry and Tetrahedralization 66

4.4.2 Juggling 67
4.4.3 Muscle Disorder 72
4.4.4 Limitations and Failure Cases 76
4.5 Discussion 77

Chapter 5 Deep Compliant Control **81**
5.1 Overview 82
5.2 Environment 84
5.2.1 Stiffness 84
5.2.2 Compliance-induced Motion Controller 86
5.3 Learning Motor Skills 89
5.3.1 Imitation Reward with Adversarial Network 90
5.3.2 State 92
5.4 Experimental Results 94
5.4.1 Environment and Controller Settings 94
5.4.2 Level of Compliance 97
5.4.3 Learning Motor Skills 98
5.4.4 Manipulation 99
5.5 Discussion 102

Chapter 6 Conclusion **104**

Bibliography **124**

요약 **125**

List of Figures

Figure 1.1 (Left) Human knee joint (Middle) Pathologic gait (Right) Jumping and landing : human senses ground reaction forces and absorb the impacts to prevent injuries when landing. 2

Figure 1.2 Actin and Myosin in muscle fibers. 3

Figure 1.3 Physics-based simulation and control of dynamic motor skills actuated by 346 musculotendon units. (Left) deadlift and cartwheel. (Middle) Pathological gait driven by hip muscle contracture. (Right) Running with a prosthetic leg. 5

Figure 1.4 Demonstration of motion control in an anatomical simulation of a juggling task, actuated by volumetric muscles. Active muscles are shown in pink. 6

Figure 1.5 Our compliant controller learns dynamic interactions. (Left to right) Hand-in-hand running, hopping game, opening a door, and balancing a ball. 8

Figure 3.1 Musculoskeletal Model. (Up right) Simplified two-toe foot. (Bottom right) Multi-segment foot. 20
Figure 3.2 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. 22
Figure 3.3 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. 24
Figure 3.4 Overview 26
Figure 3.5 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. 35
Figure 3.6 The *Deadlift* example. (Left) The number of maximally activated muscles. (Right) The height of the bar relative to simulation time. 37
Figure 3.7 Assorted motor skills: Walk, run, sargent jump, deadlift, cartwheel, and kick. 39
Figure 3.8 Walking, running, and dancing with a prosthetic leg. . . 39
Figure 3.9 Transtibial and transfemoral prostheses. 40
Figure 3.10 Pathologic Gaits. (Top to Bottom) Hip flexion contracture, tip toe, asymmetric stiff knee, and multiple symptoms. 41
Figure 3.11 Orthopedic surgery simulation. 42

Figure 4.1 (Left) The simulated muscles and skeletal components. (Middle) Low-resolution and high-resolution simulation meshes. (Right) Idealized Hill-type actuator. 49
Figure 4.2 Transmission of muscle force to bone. A rigid transformation is applied such that a vector in the direction of the central (contractile) line segment will be rotated parallel to the line segment adjacent to the insertion. . . 57
Figure 4.3 The overview of our simulation and control framework: Rigid body states set Dirichlet boundaries for muscle force computation. Our hierarchical controller takes both rigid and soft body states as input and optimizes joint trajectories and muscle activation levels. 58
Figure 4.4 Muscle modeling and rendering 65
Figure 4.5 Force-length-velocity curves. (Left) Maximal contractile force at the origin of the Biceps short head when it lengthens and shortens periodically at two different speeds. (Right) The Biceps generates larger force when it lengthens quickly. 66
Figure 4.6 Siteswap patterns of (up) 333 and (down) 423 juggling. . 68
Figure 4.7 A finite state machine for juggling and simulation parameters. 70
Figure 4.8 Juggling patterns. 71

Figure 4.9 Atrophy and Hypertropy of Biceps and Brachialis in the right arm. (Left) The crossectional area is scaled down by a factor of 0.5. (Right) The crossectional area is scaled up by a factor of 1.5. Muscle hypertrophy allows the weight (10 kg) to be lifted easily at low muscle activation levels. 72

Figure 4.10 Muscle weakness simulation. The weight of the dumbbell is 5kg. (Left) The muscles in the right arm shown in dark brown are weaker than normal ability. (Right) The scaling of the force-length curve determines the level of weakness. 73

Figure 4.11 Progressive paralysis of Biceps and Brachialis in the right arm. (From left to right) As dark cells spread, Biceps and Brachialis become weaker and therefore the nearby muscles in the forearm and the shoulder activate more to compensate for the weakness. 74

Figure 4.12 Muscle contracture and orthopedic surgery simulation. (Left) A 2D point on the texture atlas maps to a 3D point on the bones. The user can easily specify the new insertion of the muscle on the texture atlas. (Middle) The increased tension of the contracted Biceps results in the flexed elbow at the relaxed arm. (Right) The surgery simulation displaces the insertion of the Biceps from the top of the *Radius* to the bottom of the *Humerus*. As a result, the Biceps becomes a one-joint muscle. Since the surgery increases the range of motion, the elbow becomes fully extended at the relaxed arm. The simulation also confirms the side-effect that the maximum torque at the elbow becomes weaker with the displaced muscle insertion. 75

Figure 5.1 Compliance of the left hand. Due to the geometric locking, the character acts stiffly when applied force is directing to the arm while the other directions have moderate compliance. 85

Figure 5.2 System overview. 89

Figure 5.3 Expert motion distribution and the controller trajectories. The level-set represents how much the state is similar to the expert distribution. The expert trajectories are mainly located in the red area. The dotted line denotes controller trajectories while the solid line denotes trajectories of the shifted state. The shifted state is indistinguishable from the expert motion distribution. . . . 93

Figure 5.4 Levels of compliance. (Left) The ratio of contact force magnitude when the character collides with an obstacle. (Right) The ratio of control torque magnitude. Both are normalized by the values of standard PD controlled character without compliant motion control. 96

Figure 5.5 The *Trampoline* example. (Left) Minimum foot height during contact. Higher value indicates lower fluctuation of the trampoline. (Right) Total energy. 98

Figure 5.6 *Opening doors* example. (Left) The constraint force can be decomposed to orthogonal and parallel components relative to the knob, shown in blue and red arrows. (Up Right) The orthogonal force with the varying door size. (Bottom Right) The parallel force. 100

List of Tables

Table 3.1 Simulation parameters 34

Table 5.1 Task-specific motion lengths and the levels of compliance. 97

Chapter 1

Introduction

Many factors in human sensorimotor system contribute to human movements. Human motor system consists of the musculoskeletal system, where bones support the body and muscles contract and relax to move the body. Human sensory system consists of taste, vision, smell, hearing, and touch. These two systems have intrinsic mechanisms that lead unique human movements. For examples, the range of joint motion is affected by the geometric structures of musculoskeletal anatomy. The knee cap (a.k.a Patella) prevents hyperextension of the knee joint, and the elasticity of muscles, ligaments, and tendons also bounds the range of motions as well (See Figure 1.1). Muscle disorder such as contracture, weakness, and paralysis would alter overall movements and result in distinctive movement patterns of each individual. In sensory system, tactile sensors sense pressures produced by the interactions, and humans behave in a flexible manner to prevent large external forces and absorb the impacts in accordance with the tactile information.

Human movements are governed by the equations of motion, and physics

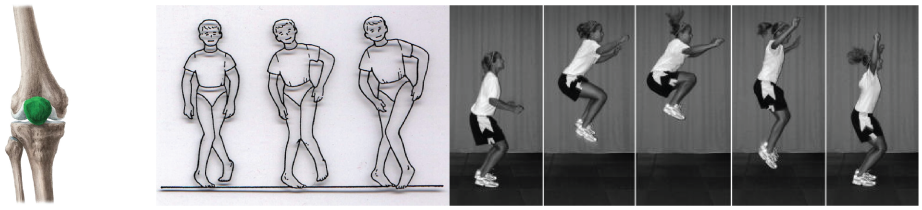


Figure 1.1: (Left) Human knee joint (Middle) Pathologic gait (Right) Jumping and landing : human senses ground reaction forces and absorb the impacts to prevent injuries when landing.

simulation has been thought to be a useful tool to understand, analyze, and reproduce human motions. Physics simulation provides in-depth observation or analysis of the way humans recognize their surroundings and actuate their bodies. When we are to use physics simulation, it is required to model digital replica of humans accurately. This means not only we need to model precise human sensorimotor system, but also we need to model suitable movement mechanisms underlying the system. Human range of motions are collaborations of bone and muscle geometry, and muscle contraction dynamics and skeleton articulations also affect the boundaries as well. Physics simulation has capacity to model and simulate such phenomenon [1].

This thesis aims to build comprehensive human sensorimotor system in physics simulation to reproduce physically reliable human movements. Fundamental challenges stem from the scalability and the complexity of human musculoskeltal system. Humans have more than 600 muscles and 200 bones, and human brain harmoniously coordinates muscle activations at every time instance to actuate its body. To simulate musculoskeletons, we need to coordinate muscle activations as well as we need accurate musculoskeletal model. It is computationally demanding to deal with 600 muscles at every time instance of physics simulation. Another challenge lies in non-linear muscle contraction

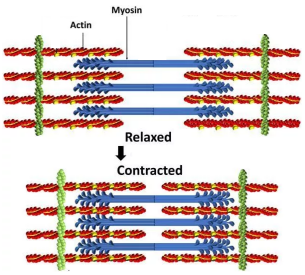


Figure 1.2: Actin and Myosin in muscle fibers.

dynamics for each muscle. A muscle consists of a bundle of muscle fibers which produce contractile forces along the fiber direction. Actin and Myosin filaments pull each other in accordance with electric signals produced by human nerve system (See Figure 1.2). The muscle also has background elasticity and the elastic forces restore muscle shapes when muscles are fully relaxed. The process of the contraction and the relaxation is highly complex and non-linear. These two characteristics of human musculoskeletal system (scalability and complexity) are major topics that we are to deal with.

While the accurate simulation of human musculoskeletal system assures physical and physiological reliability of reproduced movements, we want to add the interactivity for the physically simulated humanoids. A human interacts with its surroundings every day such as opening doors, playing sports, and pushing boxes, and the human interactivity is crucial virtue to plan such interactions as intended. In these situations, tactile sensors provide one of the primary recognition that decides the way how humans move. Tactile elements sense instantaneous pressures and humans inherently avoid large impacts that could potentially result in falling down. We often call this behavior a compliant (or flexible) behavior and it iterates a dynamic process of recognition and actuation. Reproducing compliant movements is another topic in this thesis.

Understanding relationship between human movements and the conditions has been a long standing goal in Computer Graphics, Robotics, and Biomechanics. We take into our consideration body conditions (weights, heights, and muscle conditions) and interaction conditions (compliance) to reproduce plausible motions. We demonstrate the variations in the musculoskeletal model generate reliable human movements ranging from walking, running to highly stylistic movements, even to pathologic ones as well. We also can generate a wide spectrum of compliant movements ranging from passive reactions to external physical perturbations, to active manipulations with clear intentions. We state this thesis provides frameworks that clearly bridge the intrinsic principles of movement mechanisms and human motions.

Scalable Muscle-actuated Human Simulation and Control. In Chapter 3, we discuss the scalability of the musculoskeletal simulation. Human musculoskeletal system is such a large-scale system, and there exist complex interactions between them. Due to complexity and high-dimensionality, many studies in Computer Graphics have aimed on specific body part or have used small number of muscles. In this chapter, we aim to build a comprehensive full-body musculoskeletal system that reproduces realistic human movements driven by muscle contraction dynamics. The 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. We include all the muscles and bones that contribute to joint movements. Our model has 342 muscles, 22 joints, and 56 degrees of freedom (DOFs). Our simulation system reliably deals with muscle contraction dynamics and joint range of motion induced by background elasticity of muscles. We also present a new control algorithm based on Deep Reinforcement Learning (DRL).

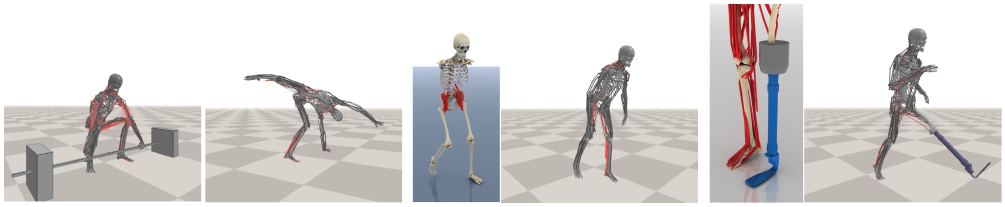


Figure 1.3: Physics-based simulation and control of dynamic motor skills actuated by 346 musculotendon units. (Left) deadlift and cartwheel. (Middle) Pathological gait driven by hip muscle contracture. (Right) Running with a prosthetic leg.

Recent advance in DRL has shown its potential in high-dimensional continuous control problem. The control policy based on deep neural networks has reproduced full-body human movements such as walking, running, cartwheel, and flipping [2]. In this case virtual humans are assumed to be actuated by the torques at each joint, and the control policy outputs the Proportional-Derivative (PD) targets which has 30 to 60 DOFs. Policy learning becomes challenging if we are to control muscle-driven humanoids, since the policy has to coordinate muscle activation levels of every musculotendon actuators. We present a new two-level control framework. Two deep neural networks hierarchically form the structure of the control policy to deal with the full-body musculoskeletal model with 346 muscles. We demonstrate the effectiveness and scalability of our framework with various examples ranging from walking, running, and cartwheel to pathological ones driven by musculoskeletal disorder, even to prostheses walking and running (See Figure 1.3).

Dexterous Manipulation and Control with Volumetric Muscles. In Chapter 4, we discuss simulation and control of musculoskeletal system with volumetric muscles. In Computer Graphics, there has been a research trail of reproducing natural human motion by incorporating the true mechanics of the

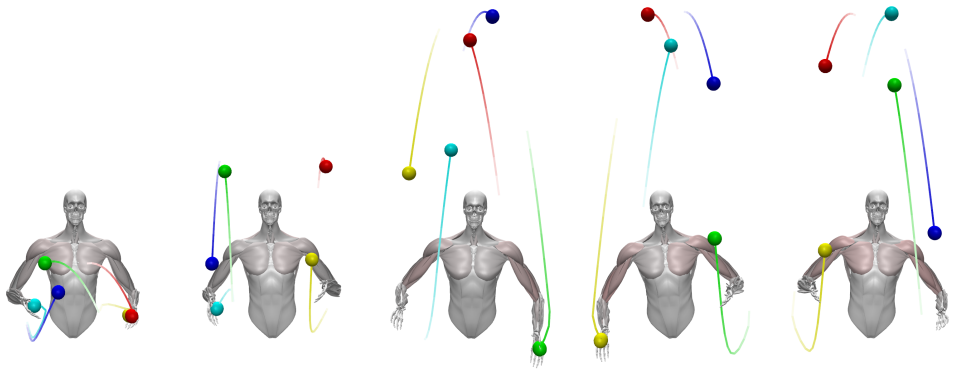


Figure 1.4: Demonstration of motion control in an anatomical simulation of a juggling task, actuated by volumetric muscles. Active muscles are shown in pink.

musculoskeletal system. However, common modeling assumptions simplify the muscle geometry to line-segment primitives, which may degrade the accuracy and the biomechanical fidelity of the simulation. Our goal is to develop a controllable system that incorporates volumetric muscles into the upper body model. We also would like to demonstrate the representation power and the functionality of the volumetric muscles. In particular, we focus on predicting physical capabilities under various body conditions.

We use Finite Element Method simulation (FEM) to simulate volumetric muscles. Since FEM is computationally expensive, it is technically challenging to incorporate volumetric simulation into muscle dynamics efficiently. Moreover, the control of humanoids with complex and non-linear muscle dynamics requires Jacobians of muscle forces with respect to control variables. Computing Jacobians, as well as integrating FEM, are extremely expensive which makes overall framework intractable. We reduce the computational costs by employing Projective Dynamics [3]. The newly designed framework for solving FEM allows

us to simulate volumetric body in real time, while we contribute a novel formulation by which active muscle forces can be accurately added to the Projective Dynamics. We also derive the Jacobians analytically by a quasistatic assumption of the simulation, which avoids numerical differentiation. The simulation framework is combined with a two-level trajectory optimization technics that controls the humanoids robustly via feed-back and feed-forward mechanisms. We demonstrate our system robustness by dexterous manipulation and control such as juggling with upper-body humanoids consisting of 128 muscle actuators (See Figure 1.4). We further examine muscle disorders such as contracture, weakness and paralysis which manifests the representation power of volumetric muscles.

Deep Compliant Control. While Chapter 4 and Chapter 3 cover the musculoskeletal simulation Chapter 5 is to reproduce dynamic process of sensorimotor control. Humans interact with their surroundings every day such as opening doors, pushing boxes, and playing sports. Forces serve as a medium for such interactions, as humans iterate sensing these forces and actuating their bodies. In this process, compliance is one of the primary principles which determines human movements during interactions. Compliance provides humans the ability to move in various ways to avoid large amounts of contact forces and adapt to unexpected situations. In this chapter, we aim to reproduce physically reliable movements for the interactions governed by compliance. Technical challenges include defining and modeling of compliance, reproducing realistic human movements, and providing robust control of motor skills for under-actuated dynamical systems. We define the character's compliance from a mechanical point of view. Our motion controller deforms the character to increase compliance. We also present a learning framework based on DRL.

Recently, DRL has been widely used to learn motor skills for high-dimensional

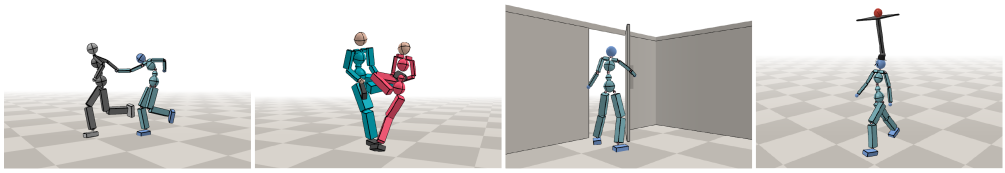


Figure 1.5: Our compliant controller learns dynamic interactions. (Left to right) Hand-in-hand running, hopping game, opening a door, and balancing a ball.

humanoid characters. The core idea in this approach stems from imitation learning where the character learns how to move from reference motion data. While existence of the reference motion data enables the imitation of human tasks, imitation learning uses position control such as PD control to generate control forces which can produce stiff and unnatural motions when external perturbations are being applied. We develop a motion planning algorithm which provides motion displacements that adjust the reference motions according to the interaction forces. The motion planner is governed by the character's compliance to reduce interaction forces as well as internal control forces. We further equip this algorithm with DRL to robustly control the characters. The resulting controller behaves in a compliant and natural way during the perturbations, and still recover its movements when there are no external forces as it regresses to conventional imitation learning. We demonstrate the effectiveness of the controller with various examples (See Figure 1.5).

Chapter 2

Background

In this chapter we briefly introduce research stream of previous work necessary for understanding this thesis. This thesis covers three topics regarding human sensorimotor system, and all these topics use physics simulation and appropriate motion control to reproduce reliable movements, and thus we divide related work into two categories: simulation and control. Since human musculoskeletal system is very complex, high-fidelity musculoskeltal simulation has been a long-standing goal in Robotics, Biomechanics, and Computer Graphics. In Section 2.1, we give you an explanation of previous work for muscle-based anatomy simulation as well as articulated skeletal simulation equipped with muscle contraction dynamics. In Section 2.2, we provide related work that have aimed on motion control ranging from trajectory optimization to DRL, and additionally, we provide studies for force control in Robotics required for sensorimotor control with tactile perceptions.

2.1 Musculoskeletal Simulation

The simulation of musculoskeletal systems has been studied for several decades [4, 5]. Ever since prototype systems have been built and released [6, 7, 8], muscle-based modeling and simulation have gained a lot of attention in both Biomechanics and Computer Graphics. A thorough survey of muscle-oriented techniques in animation tasks has also been explored [9]. Over the past two decades, Computer Graphics research has pursued higher biomechanical accuracy in anatomy modeling, using Finite Element Method simulations [10] and models derived from medical imaging [11]. The emphasis on visual plausibility instead of absolute biophysical accuracy has allowed Computer Graphics research to be more aggressive in the pursuit of full-body soft tissue simulations, especially when volumetric simulation is involved, compared to the inclination of biomechanics literature towards localized parts of anatomy (e.g. thigh mechanics [12]). Simplified line-segment muscle models have allowed anatomical modeling to address inverse problems and motion control scenarios, on which volumetric simulation is often layered as a visual embellishment [13, 14].

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 [15]. Modeling muscle fatigueness in lower body generates physiologically retargeted motions [16]. Pai and his colleagues explored the simulation of hand manipulation, including musculotendinous simulation with sliding constraints [17]. A coupled Eulerian-on-Lagrangian formulation is utilized in elastic tendon strands with a control strategy to simulate hand articulation [18]. The Eulerian-On-Lagrangian framework has been applied to the human upper extremity [19], demonstrating the ability of the framework to handle muscle contact, and coupling to an articulated skeleton. Nakada et al. [20] modeled

human sensorimotor control using deep neural networks, which emulate neural pathways from visual perception to the activation of motor units.

In the domain of facial animation, volumetric descriptions of facial muscles is used to infer activations that move the face surface in accordance with a motion-captured target [21]. Ichim et al. [22] optimize shape and contractile properties of facial tissues, starting from a template, as to reproduce input blendshape-based animations, yielding facial models that can be actuated into new animation sequences. The concept of blendshapes to material properties is extended, allowing simulated facial animations to reproduce motion effects attributable to temporally variable constitutive properties [23].

Separate from issues related to simulation methods, the authoring of simulation-ready subject-specific models is a great challenge, even if a small number of geometric templates exist (which however do not provide strict specifications of material properties, and are not subject-specific). Skeletal geometry and kinematics are jointly inferred from a temporal deformation of only the outer skin surface [24]. The way of generating a spectrum of human body types with various degrees of muscle growth is proposed, evolving from a standard template using physical processes [25]. Surface scan data from various subjects and body poses is also used to reconstruct personalized anatomical muscle models [26]. Akhter and Black [27] collected a motion capture dataset that includes a variety of stretching poses and learned a pose-dependent model of joint limits. Jiang and Liu [28] used the same dataset to learn a joint limit model represented by a fully-connected neural network.

2.1.1 Simulation of nonlinear volumetric solids

Simulation performance, which has been a significant obstacle in early efforts, has been improved by virtue of stable implicit time integration schemes [29],

accelerated deformers based on regular lattices [30], improved contact handling mechanisms [31], and improved interactivity with Projective Dynamics schemes [3]. Lattice-based deformers make simulator accommodate nontrivial topological features (e.g. surgical incisions), while safeguarding the regularity of the underlying data structures and the benefits of parallelism [32]. Hierarchies of embedded discretizations are proposed to selectively infuse dynamic degrees of freedom where artists suggest additional deformation detail is desired [33]. Data-driven frameworks enable creating dynamic deformations of flesh and tissue on articulated characters [34, 35]. A layered dynamic simulation of flesh is used, superimposed on a kinematic skeleton, to synthesize detailed skin deformation [36]. One-way or two-way coupled simulation of skeleton and flesh enables interactive, skeleton-driven animation of musculoskeleton [37, 38]. Real-time flesh simulation is also available reducing its high dimensionality in rig space [39]. Multi-body simulation systems have been developed to deal with rigid bodies, soft bodies and their coupling in a uniform manner [40, 41].

2.2 Motion Control

Controllable musculoskeletal models have overwhelmingly relied on line-segment approximations of Hill-type models to actuate an articulated character skeleton. Sueda et al. [17] introduced strand dynamics and sliding/surface constraints for hand simulation and control with emphasis on tendon dynamics. Balancing the head with neck muscles has been studied, coordinating their activations by feed forward and feed back rules [15]. Muscle activation levels in tongue can be estimated by solving quadratic programming [42].

Muscle-driven locomotion control emerged with the help of stochastic optimization methods, such as CMA-ES (Covariant Matrix Adaptation Evolution-

ary Strategy), while locomotion control problem without musculotendon actuators has been solved by a rule-based [43], learning-based [44], data-driven [45], and optimization-based [46] controller. Wang et al. [47] 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. [48] 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. [49] 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. [50].

Due to the complexity of the musculoskeletal system, the action of volumetric simulation mainly used to manifest in skeletal articulation. Control formulations have been applied to scenarios where the volumetric deformation itself is the primary output being optimized [51], or where locomotion is caused by deformation rather than skeletal articulation [52].

2.2.1 Learning Motor Skills with DRL

In 2007, the biped models had less than 10 actuated DOFs [44, 43]. The DOFs of the dynamic models increased more than tenfold by 2014 [49]. This trend still continues with the recent addition of deep network models [20]. The scalability of stochastic optimization methods does not match the exponential growth of the complexity of dynamic models and thus 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 [53, 54], dynamic sports activities [55, 56, 2, 57], exotic creatures [58, 59], and learning to dress [60]. 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 [61]. Given a reference motion, the learned control policy can reproduce high-quality human motion [2, 57]. Recently, generic neural network policies capable of learning diverse behaviors and multiple motor skills are attracting attention [62, 63, 64]. Not only human-like characters, but also flying creatures [65], quadrupeds [66] and even underwater animals like octopuses [67] are succesfully controlled with learned policy with DRL.

Recent approaches exploiting hierarchical structures on DRL have shown their potential to improve the scalability of control systems. Levy et al. [68] used multi-level hierarchies to accommodate sparse reward tasks. Vezhnevets et al. [69] 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. [70] 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 [71, 72, 73]. 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.

2.2.2 Force Control

Force control is one of the major topics in robot control and has been thoroughly studied in Robotics over the past four decades. While position-based control such as PD control focuses mainly on robot positioning, force control aims to achieve precise control of the forces applied by end-effectors. The concepts of impedance and admittance are terminologies used to describe the relationship between the manipulator and the environment [74, 75]. A manipulator is considered an admittance when it controls motions to minimize the external forces, whereas a manipulator is considered an impedance when it controls the forces from the motions. Our framework comprises both the impedance control and the admittance control.

One way to minimize the interaction forces is to embed spring-actuated joints so that the joint has inherent elasticity[76, 77]. Another way is to design a controller as a virtual spring-damper system where the virtual forces of the spring and the damper act as elastic actuators [78, 79]. The control module can be applied to many applications where safety is crucial such as exoskeletal robot control [80, 81, 82, 83] and control in contact-rich spaces [84, 85]. Recent advances in deep learning have enriched the domain of force control. Learning of vision and haptic feedback generates robot trajectories for contact-rich tasks in unstructured environments [86]. Despite great successes in force control, studies have mainly focused on fully actuated robots with few rigid links. In this case, there exist closed-form solutions obtainable by solving the differential kinematics, and it is straightforward to compute trajectories for compliance. It is not straightforward to apply control methods to under-actuated characters. We present a two-level architecture to accommodate the bipedal character.

Meanwhile, there has been a series of studies in Computer Graphics mim-

icking the force control to reproduce human reactions. Data-driven approaches with semi-physics allow the character to react to external forces such as hitting or pushing [87, 88]. Push-and-recovery tests are conducted to analyze the stability of the locomotion controller [89, 90]. Curriculum learning with DRL can be used to generate task-specific human responses [91]. Our framework addresses human reactions in terms of compliance and demonstrates various examples that reproduce physically reliable movements.

Chapter 3

Scalable Muscle-Actuated Human Simulation and Control

Human musculoskeletal system is such a large-scale system consisting of more than 600 muscles and 200 bones. Due to its large-scale, simulating full-body musculoskeletal system is especially challenging. This chapter aims to build a comprehensive musculoskeletal model and its control system that reproduces realistic full-body 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.

3.1 Overview

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 underactuated 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 [2]. It has been reported that the performance of DRL for continuous control depends on the choice of actuator types [72]. 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 [71]. 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 ortho-

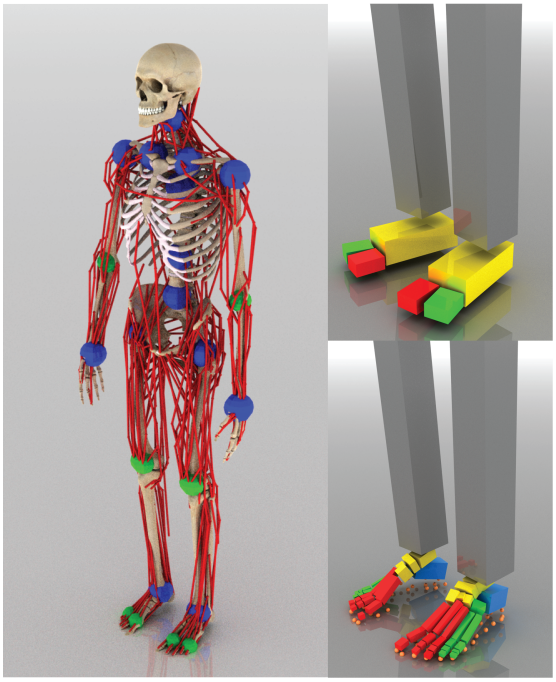


Figure 3.1: Musculoskeletal Model. (Up right) Simplified two-toe foot. (Bottom right) Multi-segment foot.

pedic 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.

3.2 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 3.1). 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.

3.2.1 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 3.2). The location of a waypoint is expressed by using the linear blending skinning (LBS) function.

$$\mathbf{p} = \sum w_j \mathbf{T}_j \mathbf{x}_j, \tag{3.1}$$

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 $a \in [0, 1]$.

$$f = f(l, \dot{l}, a) = a \cdot f_l(l) \cdot f_v(\dot{l}) + f_p(l), \tag{3.2}$$

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,

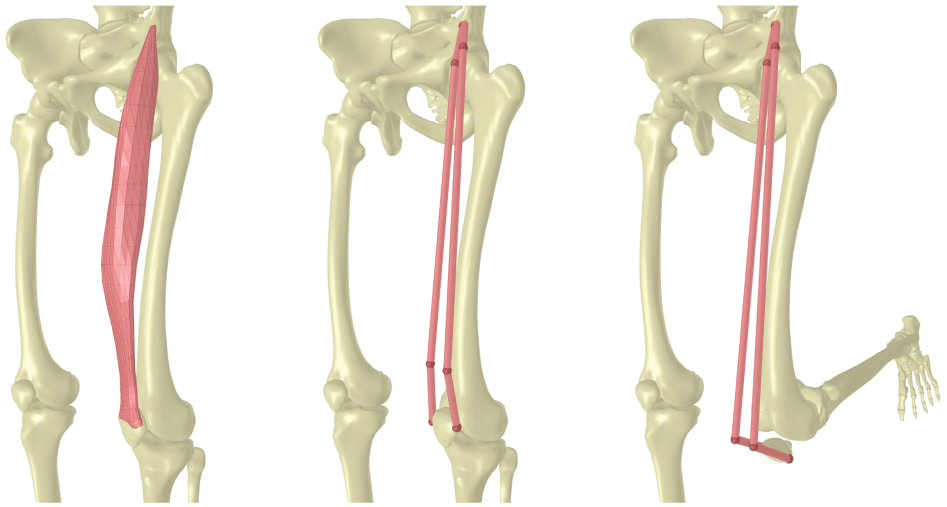


Figure 3.2: Rectus Femoris muscle. (Left) 3D surface mesh. (Middle) Polyline approximation with waypoints. (Right) The LBS coordinates of the waypoints approximate the muscle route when the knee flexes.

the muscle develops passive force $f_p(l)$ because of its background elasticity. We refer the reader to the work by Thelen [5] and Delp et al. [6] 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 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 $\mathbf{f}_m(a)$.

3.2.2 Muscle-induced Joint limit

The range of joint motion is affected by many factors including bone, joint, and 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 3.3). 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.

$$C_m(\mathbf{q}) := f_p^{\max} - f_p(l(\mathbf{q})) \geq 0 \tag{3.3}$$

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:

$$\mathbf{v}_{\mathrm{c}} := \frac{d\mathbf{C}}{dt} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} = \mathbf{J}_{\mathrm{c}} \dot{\mathbf{q}} \tag{3.4}$$

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

Given the constraint equation $\mathbf{C}$ and its velocity $\mathbf{v}_{\mathrm{c}}$, the constraint force $\mathbf{f}_{\mathrm{c}}$ that enforces the constraint can be computed by impulse-space simulation [92], 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:

$$\begin{aligned} \mathbf{v}_{\mathrm{c}} &\geq 0 \\ \mathbf{f}_{\mathrm{c}} &\geq 0 \\ \mathbf{v}_{\mathrm{c}}^{\top} \mathbf{f}_{\mathrm{c}} &= 0 \end{aligned} \tag{3.5}$$

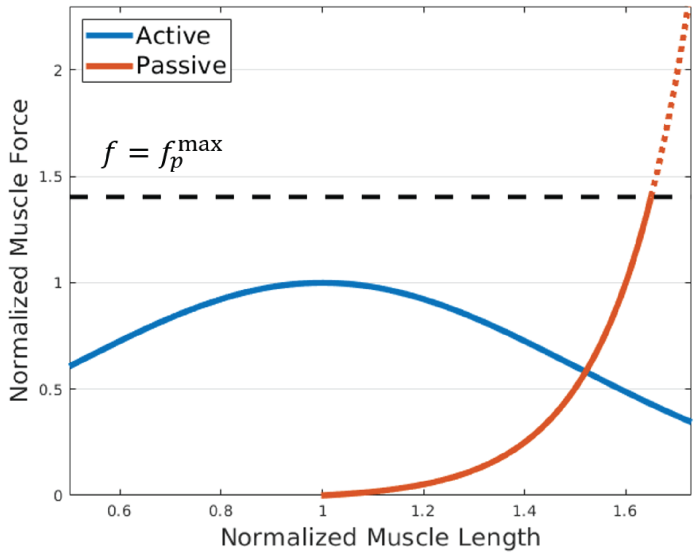


Figure 3.3: Force-length curves in Hill-type muscles. (Blue) Maximum isometric force by active contraction of muscle fibers. (Red) Passive muscle-tendon tension when the fibers are inactive.

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.

3.2.3 Musculoskeletal Dynamics

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

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_m \mathbf{J}_m^\top \mathbf{f}_m(a_m) + \mathbf{J}_\mathrm{c}^\top \mathbf{f}_\mathrm{c} + \tau_\mathrm{ext} \tag{3.6}$$

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 $\mathbf{f}_\mathrm{c}$ are muscle and constraint forces, respectively. $\mathbf{J}_m$ and $\mathbf{J}_\mathrm{c}$ 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 (4.10) 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:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{c} = \mathbf{A}\mathbf{a} + \mathbf{p} + \mathbf{J}_\mathrm{c}^\top \mathbf{f}_\mathrm{c} + \tau_\mathrm{ext} \tag{3.7}$$

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 $\mathbf{a}$.

$$\ddot{\mathbf{q}} = \mathbf{L}\mathbf{a} + \mathbf{b} \tag{3.8}$$

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

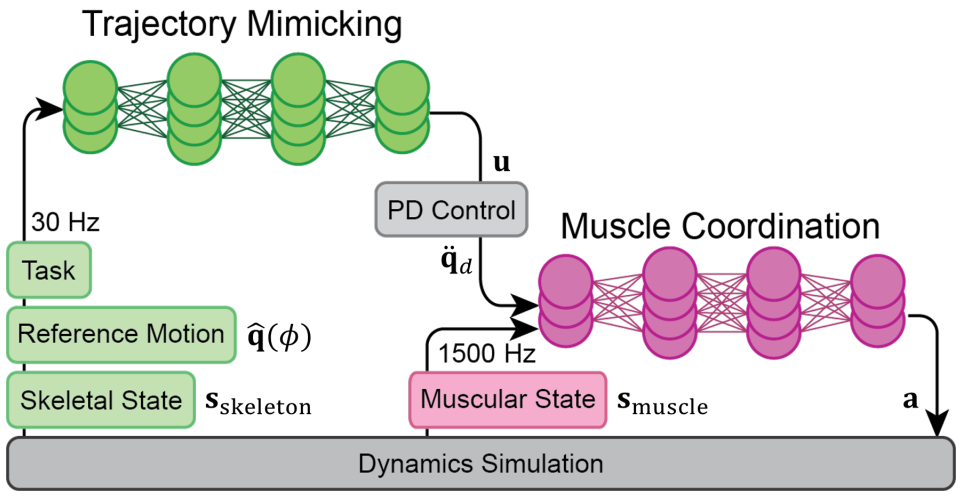


Figure 3.4: Overview

3.3 Control

To animate musculoskeletal models in physics-based simulation environments, we follow the trail of data-driven (a.k.a. example-guided) approaches [49, 2]. 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 5.2). 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.

3.3.1 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:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\mathbf{s}_0, \mathbf{u}_0, \mathbf{s}_1, \cdots} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t) \right] \tag{3.9}$$

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 [93, 94, 95]. We largely follows the implementation by Peng et al. [2] 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.

$$\ddot{\mathbf{q}}_{\text{d}}(\mathbf{u}) = k_{\text{p}}(\mathbf{u} - \mathbf{q}) - k_{\text{v}}\dot{\mathbf{q}} \tag{3.10}$$

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

$$r = w_q r_q r_e + w_g r_g \tag{3.11}$$

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.

$$\begin{aligned} r_q &= \exp\big(-\sigma_q \sum_j \|\hat{\mathbf{q}}_j(\phi) \ominus \mathbf{q}_j\|^2\big) \\ r_e &= \exp\big(-\sigma_e \sum_e \|\hat{\mathbf{p}}_e(\phi) - \mathbf{p}_e\|^2\big) \end{aligned} \tag{3.12}$$

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 $\mathbf{q}_1 \ominus \mathbf{q}_2 = \ln(\mathbf{q}_2^{-1}\mathbf{q}_1)$ [96]. 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.

3.3.2 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.

$$\begin{aligned} \min_{\mathbf{a}} \quad & \|\ddot{\mathbf{q}}_{\mathrm{d}}(\mathbf{u}) - \ddot{\mathbf{q}}(\mathbf{a})\|^2 + w_{\mathrm{reg}}\|\mathbf{a}\|^2 \\ \text{subject to} \quad & \mathbf{M}\ddot{\mathbf{q}} + \mathbf{c} = \sum_m \mathbf{J}_m^\top \mathbf{f}_m(a_m) + \mathbf{J}_{\mathrm{c}}^\top \mathbf{f}_{\mathrm{c}} + \tau_{\mathrm{ext}} \\ & 0 \le a_m \le 1 \quad \text{for} \quad \forall m. \end{aligned} \tag{3.13}$$

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 (3.8). Incorporating Equation (3.8) into Equation (4.14), the loss function for training π_ψ is:

$$\text{Loss}(\mathbf{a}(\psi)) = \mathbb{E}\big[\|\ddot{\mathbf{q}}_{\text{d}}(\mathbf{u}) - \mathbf{L}\mathbf{a}(\psi) - \mathbf{b}\|^2 + w_{\text{reg}}\|\mathbf{a}(\psi)\|^2\big] \tag{3.14}$$

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 $(\mathbf{L}, \mathbf{b}, \mathbf{u}, \text{vec}(\mathbf{A}), \mathbf{p})$. 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.

3.4 Experimental Results

3.4.1 Simulation Settings and Muscle Parameters

Our dynamic simulation is written in C++. Open source library DART is used to simulate skeletal dynamics [97]. 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 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 [56]. 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 3.1).

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 [98] to set the initial values of physical parameters and also tuned the values further to make the model viable for physics-based simulation.

3.4.2 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 [99]. Our trajectory mimicking policy is trained using PPO [95]. 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

Table 3.1: 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

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

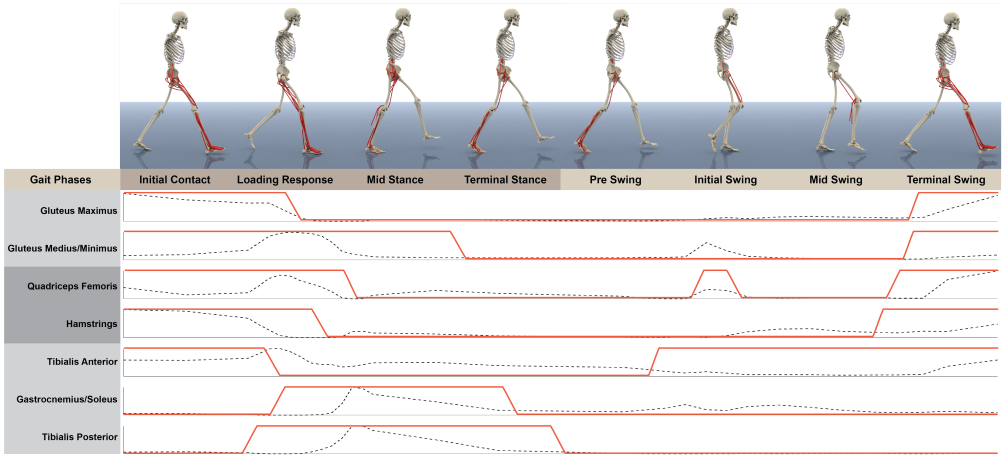


Figure 3.5: The eight phases of gait cycle. Activation levels at lower-limb muscles are plotted in black dotted lines. The EMG reference data are shown in red solid lines.

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.

3.4.3 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 3.1 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 [100]. We compare our simulation results with the reference electromyography (EMG) data during gait cycle in Figure 3.5. 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:

$$r_g(\mathbf{s}) = e^{-\gamma(\hat{y}_{\mathrm{COM}} - y_{\mathrm{COM}}(\mathbf{s}))^2} + e^{-\gamma(\hat{y}_{\mathrm{lf}} - y_{\mathrm{lf}}(\mathbf{s}))^2} + e^{-\gamma(\hat{y}_{\mathrm{rf}} - y_{\mathrm{rf}}(\mathbf{s}))^2} \tag{3.15}$$

where $\hat{y}_{\mathrm{COM}}$, $\hat{y}_{\mathrm{lf}}$, and $\hat{y}_{\mathrm{rf}}$ 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 $\hat{y}_{\mathrm{COM}} = 1.3$ meter and $\hat{y}_{\mathrm{lf}} = \hat{y}_{\mathrm{rf}} = 1.0$ 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

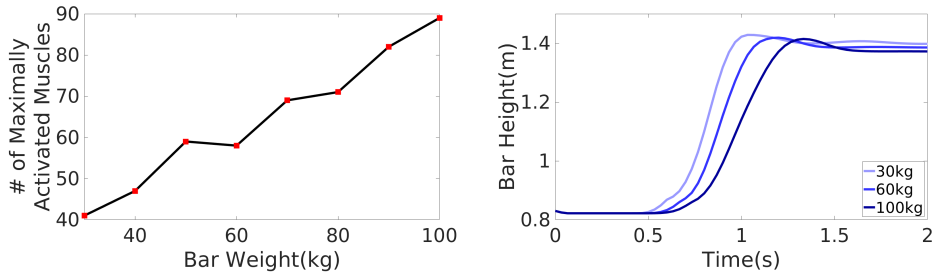


Figure 3.6: The *Deadlift* example. (Left) The number of maximally activated muscles. (Right) The height of the bar relative to simulation time.

is no improvement in target height over one 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 3.6). 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.

$$r_g(\mathbf{s}) = e^{-\gamma||\mathbf{p}_{\text{left}} - \mathbf{p}_{\text{right}}||^2} \tag{3.16}$$

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 3.8). 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 3.9). 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.

3.4.4 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, with severe conditions, the reference

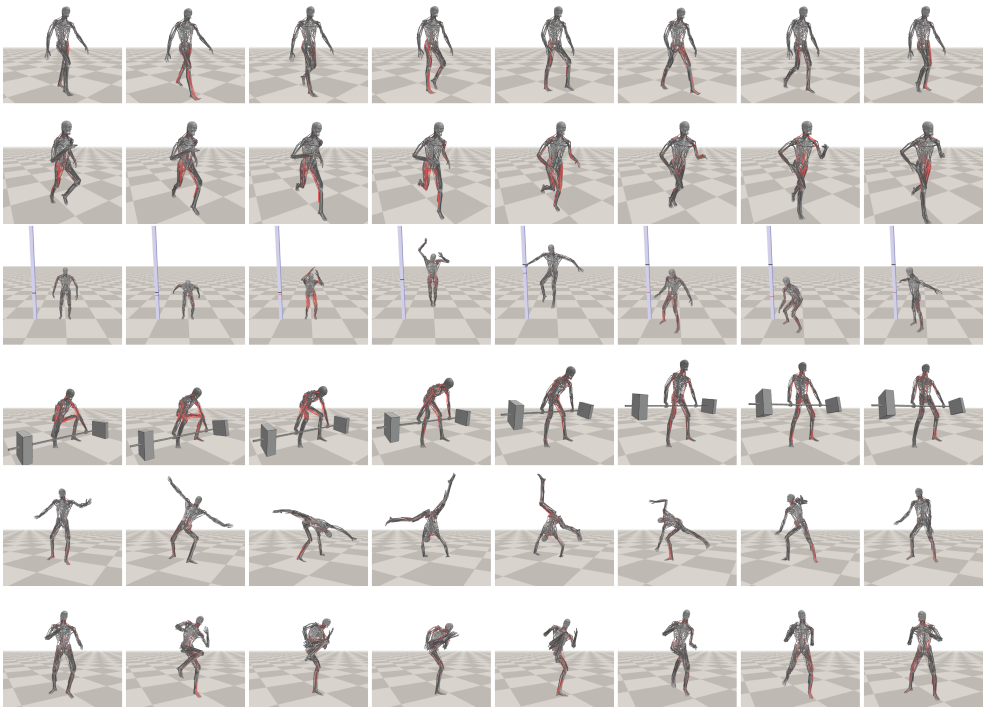


Figure 3.7: Assorted motor skills: Walk, run, sargent jump, deadlift, cartwheel, and kick.

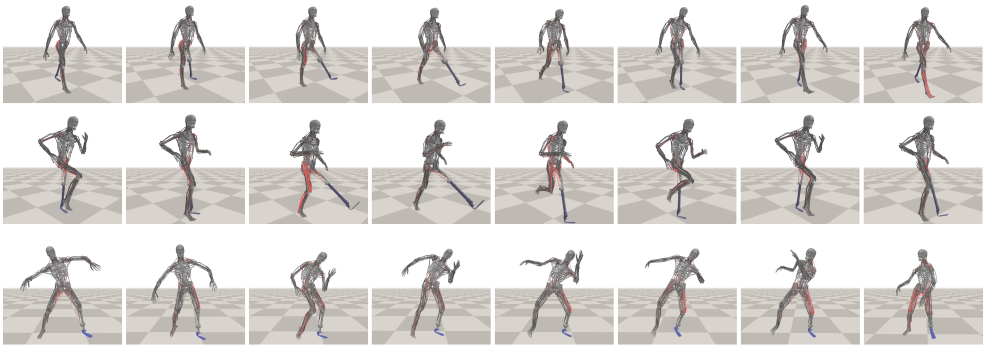


Figure 3.8: Walking, running, and dancing with a prosthetic leg.

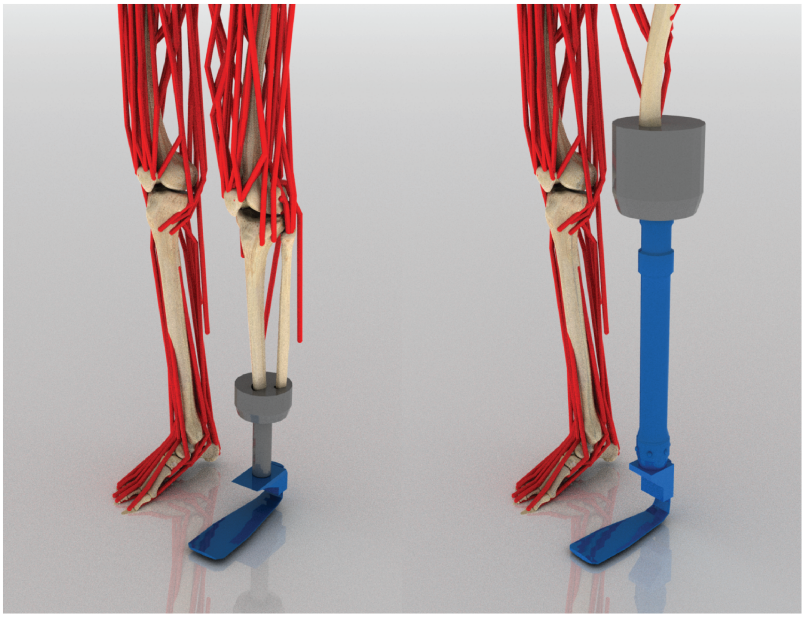


Figure 3.9: Transtibial and transfemoral prostheses.

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.

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

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 mo-

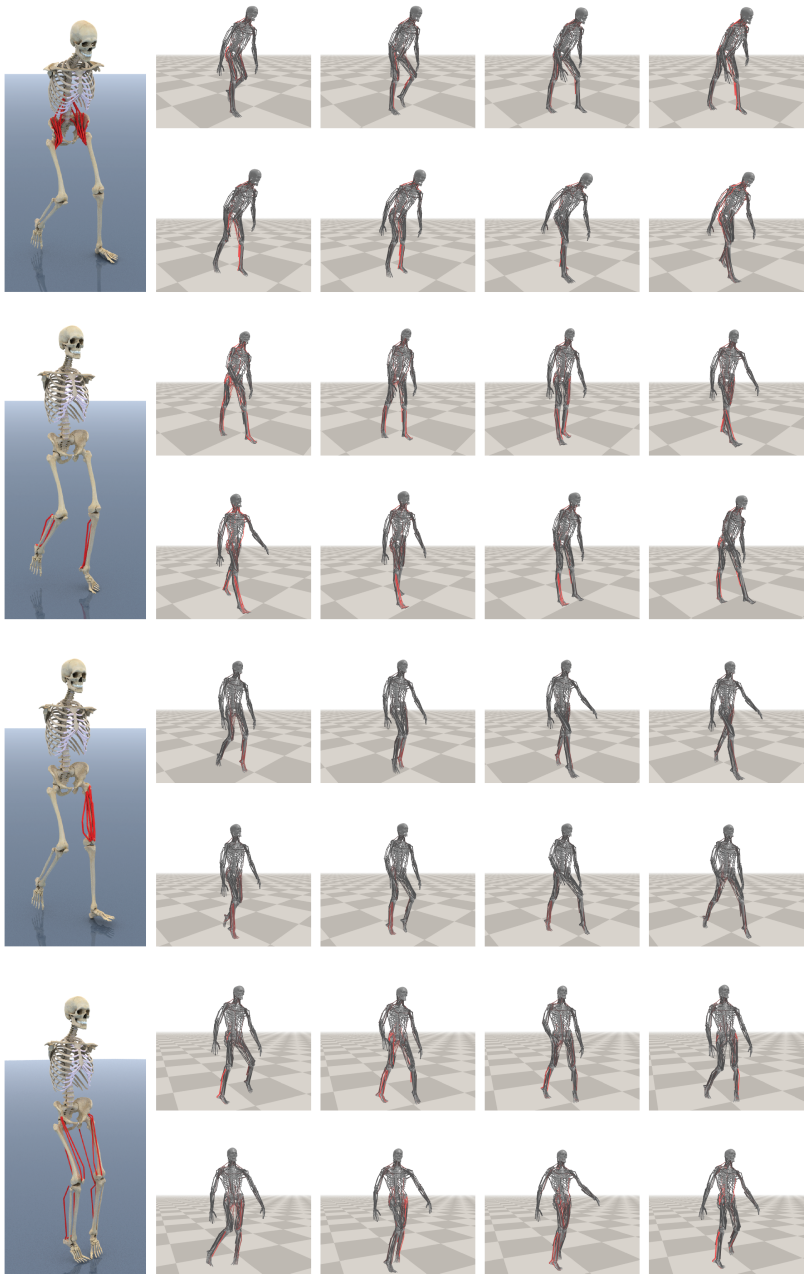


Figure 3.10: Pathologic Gaits. (Top to Bottom) Hip flexion contracture, tip toe, asymmetric stiff knee, and multiple symptoms.

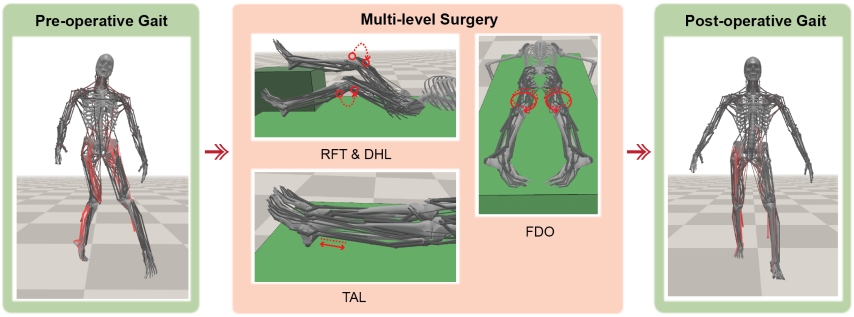


Figure 3.11: Orthopedic surgery simulation.

tion. We used IPOPT [101] to solve for the frames.

With the aid of this optimization, we successfully learned four types of pathologic gaits (see Figure 3.10). 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 3.11). 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.

3.5 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.

Chapter 4

Dexterous Manipulation and Control with Volumetric Muscles

As we demonstrate simulation framework for full-body musculoskeletal model, we simplify the muscle geometry and its contraction dynamics with line-segment primitives and closed-form arithmetic force-curve functions respectively. This approximation implies that the representation of the muscle dynamics is limited to express anatomical facts such as muscle-muscle contact and cross sectional muscle contractile forces. Such anatomical features relate to muscle geometries, and we need to define the muscle model in details. In enhancement of line-segment approximations that prior work is overwhelmingly restricted to, we incorporate volumetric muscle actuators into our framework. This drastically increases the dimensionality of our framework, and it is computationally demanding to simulate volumetric muscles. In this chapter we discuss efficient simulation for the volumetric muscles that are tightly coupled with a control system capable of demonstrating complex motion tasks such as juggling, and weightlifting sequences with variable anatomic parameters and interaction con-

straints.

4.1 Overview

Complex movement of a human body emerges from the geometrical structure of the musculoskeletal system and its mechanical characteristics. The skeleton supports the body, and the muscles contract to induce motion of the attached bones. The human brain coordinates the activations of the muscles in harmonious synergy, to create intended bulk motion or maintain kinematic balance. In Computer Graphics, there is a strong research trail of methods for reproducing natural human motion by incorporating the true mechanics of the musculoskeletal system. However, common modeling assumptions such as the near-ubiquitous adoption of line-segment primitives for the actuation of such systems raise a number of important questions: First, it is not well understood what impact such simplifying assumptions may have on the accuracy and biomechanical fidelity of the simulations thus produced. Second, it is unclear if we could forego such simplification by just accepting an increase in the computational cost, or whether current control formulations would be challenged to accommodate more complex, volumetric actuators. Finally, it is reasonable to question whether the incorporation of volumetric muscle primitives with control techniques can cope with the complexity of regenerating highly coordinated dexterous skills.

The Hill-type muscle model [4, 5] has been broadly adopted to encode the nonlinear contraction dynamics of muscle in fields such as ergonomics, computer graphics, and robotics. However, almost every prior attempt at *tightly coupled simulation and control* of the musculoskeletal system has resorted to simplifying muscles into sequences of line segments (some of which are capable

of active contraction), consciously neglecting the geometrical structure of volumetric muscles (e.g. cross-sectional geometry and active fiber field distribution). In addition, various physical parameters such as pennation angle, maximum force, and way-points in the Hill-type muscle model have to be tuned according to the actual muscles, in order to empirically match the physiological behavior of observed human motion.

Prior work has progressed towards tightly integrated volumetric musculature simulation and control, but has stopped just short of achieving this goal. Lee and his colleagues demonstrated a volumetric musculoskeletal simulation, but derived bone kinematics and muscle activations from a separate line-segment simulation and controller [13]. A volumetric simulation is primarily used for visualization and to mediate force transfer from an aquatic environment to the skeleton, while the controller still uses line-segment muscles [14]. Fan and his colleagues demonstrates the impressive accomplishment of stable articulation of the skeleton based on the action of volumetric primitives; nevertheless, control is consciously left outside the scope of this work [19]. Volumetric muscles have been used in tight integration with control in the realm of facial animation [21], but without directly articulating the mandible, nor including any other skeletal bones. We claim that this paper presents the first work in musculoskeletal simulation in the domain of computer graphics that successfully incorporates volumetric muscles in tight integration with a motion controller.

The Finite Element Method (FEM) has been used in a large segment of prior work on anatomical simulation, and it affords a broad spectrum of constitutive models to convey the mechanics of biomaterials. In our work, however, we use a Corotated Elasticity formulation for the background isotropic elasticity of muscle tissue [29, 25], which allows us to employ Projective Dynamics [3] for improved robustness and performance. In addition, we contribute a novel

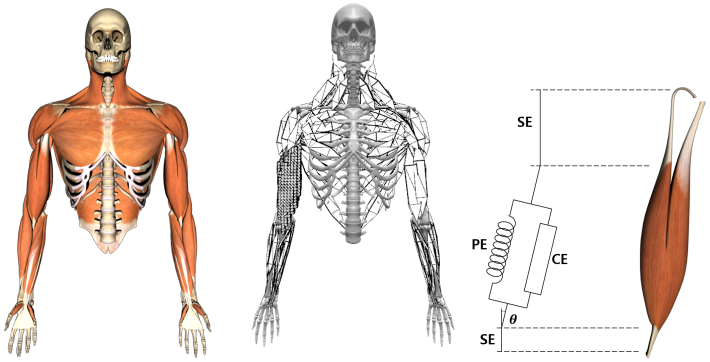


Figure 4.1: (Left) The simulated muscles and skeletal components. (Middle) Low-resolution and high-resolution simulation meshes. (Right) Idealized Hill-type actuator.

formulation by which the active Hill-type muscle force can be accurately added to the Projective Dynamics framework, while retaining the robustness and efficiency of the method. We demonstrate how Jacobians of muscle forces, required for optimization-based motion controllers, can be analytically computed via a quasistatic assumption of the muscle deformation. The simulation framework is combined with a two-level trajectory optimization approach, motivated by the formulation of Lee et al. [49] and adapted to the intricacies of our volumetric actuators.

4.2 Human model

Our model focuses on the upper body musculoskeletal system. We use an articulated skeleton with 19 degrees of freedom, and 130 motor units (i.e. independently activated contractile regions within muscles), as illustrated in Figure 4.1(left). The skeleton is actuated as a result of muscle excitation, which is respectively governed by nonlinear internal dynamics. We simulate volumetric

muscle volumes, discretized into individual tetrahedral meshes, with constitutive properties following a Hill-type model [4]. Sections 4.2.1 and 4.2.2 detail the volumetric muscle model discretization and simulation, Section 4.2.3 describes our modeling of the skeletal system, while coupling between the two is detailed in Section 4.2.4.

4.2.1 FEM simulation

We assume tetrahedral meshes of muscles provided as input. Since we individually simulate each muscle volume (which might however contain several independently controllable contractile regions), consider any individual one of these meshes with k discrete vertices. The state of this muscle model is represented in the nodal positions $\mathbf{x}_n \in \mathbb{R}^{3k}$ and velocities $\mathbf{v}_n \in \mathbb{R}^{3k}$, where the subscript indicates a time instance t_n in the dynamic evolution of this model. Integrating the equations of motion in accordance with a Backward Euler scheme, provides the following update rule:

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h\mathbf{v}_{n+1} \\ \mathbf{v}_{n+1} &= \mathbf{v}_n + h\mathbf{M}^{-1}(\mathbf{f}_{\text{int}}(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}}) \end{aligned} \tag{4.1}$$

where h denotes the length of the time step, $\mathbf{M} \in \mathbb{R}^{3k \times 3k}$ is the mass matrix, and $\mathbf{f}_{\text{int}}(\mathbf{x}) = -\nabla_{\mathbf{x}} E(\mathbf{x})$ are internal forces computed from an elastic strain energy $E(\mathbf{x})$, and $\mathbf{f}_{\text{ext}}$ is the external force. We note that, in principle, the internal forces could also be dependent on the nodal velocities ($\mathbf{v}$) as well; this could be the case if explicit damping was incorporated into our methodology, or when material properties are allowed to vary as a function of velocity. We do not include explicit damping in our simulation, but there are velocity-dependent material properties stemming from the force-velocity relationship of Hill-type muscles as discussed in Section 4.2.2. In order to simplify our evolution, we use

the velocity computed at the end of the previous time step ($\mathbf{v}_n$) whenever this appears in the expression of equation (4.1), essentially rendering it a constant for the purposes of the evolution from time t_n to t_{n+1}; hence, for simplicity we retain the notation $\mathbf{f}_{\text{int}}(\mathbf{x}_{n+1})$ for the forces in the update equation, indicating that only the positions $\mathbf{x}_{n+1}$ are to be solved for.

In general, the Backward Euler update is solved by using a Taylor expansion to approximate the nonlinear force as $\mathbf{f}(\mathbf{x}_{n+1}) \simeq \mathbf{f}(\mathbf{x}_n) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\big|_{\mathbf{x}_n} (\mathbf{x}_{n+1} - \mathbf{x}_n)$. This approximation yields the following linear system, which we alternate between solving and updating the linearization, until convergence:

$$\left[\mathbf{M} - h^2 \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right] \mathbf{v}_{n+1} = \mathbf{M}\mathbf{v}_n + h(\mathbf{f}_{\text{int}}(\mathbf{x}_n) + \mathbf{f}_{\text{ext}}) \tag{4.2}$$

In order to reduce the computation cost and improve the robustness of the Newton approach, we adopt the Projective Dynamics formulation [3], who solve an equivalent problem by alternating an efficient and parallelizable *local* update rule, with a *global* purely quadratic problem, which can be accelerated by pre-factorizing its Hessian as a one-time cost. The Projective Dynamics formulation is centered on the premise that certain elastic strain energies can be written as the sum $E(\mathbf{x}) = \sum_i E_i(\mathbf{x})$, where each term $E_i(\mathbf{x})$ (conceptually associated with an individual *constraint*) has the form:

$$\mathbf{E}_i(\mathbf{x}) = \min_{\mathbf{p}_i \in C_i} \frac{k}{2} \|\mathbf{A}_i \mathbf{x} - \mathbf{p}_i\|^2 \tag{4.3}$$

Here, C_i is a manifold associated with each individual constraint, and $\mathbf{p}_i$ is the "projection" of the quantity $\mathbf{A}_i\mathbf{x}$ onto this manifold, which makes this energy match the expression of the original strain energy, once the solution has been reached. The definition of $\mathbf{A}_i$ and the process for computing the projection $\mathbf{p}_i$ is problem-dependent; for details we refer to the original work of [3]. The local step amounts to updating all $\mathbf{p}_i$ values by projecting $\mathbf{A}_i\mathbf{x}$ onto each respective

C_i, and in the global step we solve the pure quadratic system in equation (4.3) by assuming all $\mathbf{p}_i$ to be held constant to their computed projections. For our problem, this global step amounts to the solution of the equation:

$$(\mathbf{M} + h^2\mathbf{L})\mathbf{x}_{n+1} = \mathbf{M}(\mathbf{x}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}) + \mathbf{Jd} \tag{4.4}$$

where $\mathbf{L} = \sum_i \mathbf{A}_i^T\mathbf{A}_i$, $\mathbf{J} = \sum_i \mathbf{A}_i^T\mathbf{S}_i$, $\mathbf{d} = \sum_i \mathbf{S}_i^T\mathbf{p}_i$, and $\mathbf{S}_i$ are selector matrices, such that $\mathbf{p}_i = \mathbf{S}_i\mathbf{d}$.

Since the matrix of this system is constant we use Cholesky decomposition to factorize it as a pre-processing step. Furthermore we adopted a quasistatic assumption which is fast and robust, by taking the limit of this equation as $h \to \infty$. Although this simplification eschews dynamic effects (e.g., jiggling), we found it to have minimal impact on our controller, partially due to the fact that muscles are attached to the bones tightly, reducing the effect of the inertial motion on simulation. Under the quasistatic hypothesis, equation (4.4) simplifies to:

$$\mathbf{L}\mathbf{x}_{n+1} = \mathbf{f}_{\text{ext}} + \mathbf{Jd} \tag{4.5}$$

4.2.2 Muscle Model

In contraction dynamics of Hill-type muscles, the muscles are divided into three parts according to their role, as seen in Figure 4.1(right). Those three parts are: the Passive Element (PE) modeling the background elasticity of the muscle, the Contractile Element (CE) generating the force when the muscle excites, and the Serial Element (SE) modeling the tendon which transfers the muscle-generated force to the skeleton. We discuss how each component is accounted for in our simulation framework.

Passive Element

We use a simple Corotational Elastic energy for modeling the background isotropic elasticity of each muscle:

$$\Psi_{\mathrm{PE}}(\mathbf{F}) = \frac{1}{2}\mu\|\mathbf{F} - \mathbf{R}\|^2 \tag{4.6}$$

where μ is the Young's modulus , $\mathbf{F} = \mathbf{U\Sigma V}^T$ is the deformation gradient for the element, and $\mathbf{R} = \mathbf{UV}^T$. This is the exact constitutive model that the Projective Dynamics formulation is centered around [3]. Equation (4.6) provides the energy for each tetrahedral element, and each such tetrahedron gives rise to one *constraint* C_i in the Projective Dynamics formulation, where $\mathbf{A}_i$ is the linear operator that maps nodal positions $\mathbf{x}$ to the deformation gradient $\mathbf{F}_i$ of the i-th element, and the projection operation maps $\mathbf{F}_i = \mathbf{U\Sigma V}^T$ to its rotational component $\mathbf{p}_i = \mathbf{R} = \mathbf{UV}^T$. Incorporating volume preservation into the formulation is mathematically involved. We refer to their paper [3] for volume preservation and details on the relevant algebra.

Contractile Element

Physiologically, when the excitation signal is delivered to the muscle, actin and myosin fibers pull each other and contract the muscle. In accordance with the approach in prior Finite Element muscle modeling approaches [11, 13], we model the anisotropic action of the contractile element by an additive contribution to the strain energy $\Psi_m(l)$, which is taken to be dependent only on the *fiber stretch factor* $l = \|\mathbf{Fd}\|$, where $\mathbf{F}$ is the deformation gradient and $\mathbf{d}$ is a unit vector in the direction of the muscle fiber. An explicit expression for $\Psi_m(l)$ is typically never referenced or pursued, as only the gradient and Hessian of this energy will ever be used in an FEM simulation. The *derivative* of this

quantity, however, is the fiber tension which is given by the well-known Hill-curve $\partial\Psi_m/\partial l := f_{hill}(l, \dot{l}, a)$ and this expression is the one actually used in simulation, where a is the level of muscle activation. An application of the chain rule provides the following formula for the Piola-Kirchhoff stress:

$$\mathbf{P}_m := \frac{\partial \Psi_m}{\partial \mathbf{F}} = \frac{\partial \Psi_m}{\partial l} \cdot \frac{\partial l}{\partial \mathbf{F}} = f_{hill}(l, \dot{l}, a) \cdot \frac{1}{l} \mathbf{F}\mathbf{d}\mathbf{d}^T \tag{4.7}$$

from which nodal forces can be readily computed. Sifakis and Barbic provide the relevant details for tetrahedral meshes [102]. It is well understood that the Projective Dynamics formulation of [3] supports a specific and somewhat narrow scope of materials within its core formulation, due to the requirement that the energy being minimized must be expressible in the form of equation (4.3). This narrow scope motivated the later approach of Liu et al. [103] that provided the opportunity to accommodate a broader gamut of materials, with some modest compromises in the robustness and efficiency of the prior formulation [3] (e.g., the need to incorporate a line search for stability).

We introduce a novel approach to incorporate the Hill-type muscle force into the exact formulation mandated by the Projective Dynamics framework. We do so by defining the following energy associated with each muscle, in the exact fashion of equation (4.3):

$$\Psi_{\mathrm{CE}}(\mathbf{F}) = \frac{1}{2} k \|\mathbf{F}\mathbf{d} - \mathbf{p}(l, \dot{l}, a)\|^2 \tag{4.8}$$

where k is a stiffness coefficient. In the local step of Projective Dynamics, the vector $\mathbf{p}(l, \dot{l}, a)$ is chosen in the subspace of vectors parallel to $\mathbf{F}\mathbf{d}$, with the specific scale factor in the expression below:

$$\mathbf{p}(l, \dot{l}, a) = \left[1 - \frac{f_{hill}(l, \dot{l}, a)}{k \cdot l}\right] \mathbf{F}\mathbf{d} \tag{4.9}$$

In the provided Appendix, we demonstrate that with the selection of this special value of $\mathbf{p}$, the Piola-Kirchhoff stress $\partial\Psi_{CE}/\partial\mathbf{F}$ computed by this expression

matches exactly the value of $\mathbf{P}_m$ in equation (4.7) corresponding to the standard active muscle stress of prior FEM approaches; thus, at equilibrium this Projective Dynamics formulation reproduces exactly the Hill-type muscle force that prior FEM muscle simulation formulations employed. With $\mathbf{p} = \mathbf{p}(l, \dot{l}, a)$ taken to be constant (in the global step), the expression $\Psi_{\mathrm{CE}}(\mathbf{x})$ becomes a pure quadratic ($\mathbf{F}$ is a linear function of nodal positions $\mathbf{x}$, as mentioned in the treatment of the corotated energy), and the Projective Dynamics formulation becomes fully applicable. We ultimately compute nodal forces via the Piola-Kirchhoff stress in equation (4.7), and distribute them to the forces of each simulated tetrahedron [102]. Note that the choice of the coefficient k in equation (4.8) is arbitrary; however, for adequately large values of k, the provided expression for $\mathbf{p}(l, \dot{l}, a)$ most closely approximates a true Euclidean projection in the vector subspace spanned by $\mathbf{Fd}$. We found that a value of $k = 10^7$ (compare with $\mu = 5 \cdot 10^6$) generated robust convergence in our examples.

Serial Element

The Serial Element models the tendons on either side of the contractile segment of the muscle, and transmits the force to the bone insertion. Physiologically, the tendon is very stiff compared to the muscle and sustains very minimal elongation even under full muscle activation. Therefore, it was our design decision to model the tendon as a taut, inextensible wire that provides the boundary condition for the volumetric muscle simulation, by following the path of the conventional piecewise line-segment muscle primitive, from the insertion and through any way-points, until its reference length has been traversed. The endpoint of the tendon thus routed provides the Dirichlet boundary condition for our volumetric muscle simulation.

We also use this idealization of the tendon as an inextensible wire, which

is routed through the muscle way-points to transmit the force generated by the volumetric muscle to the attached bones. We do so by rigidly transforming (i.e. rotation) the force along the way-points. Focusing on the way-points of origin of ith muscle, the axis of wire between $(j-1)$th and jth way-point can be defined as $\mathbf{u}_j = \frac{\mathbf{p}_{j-1}-\mathbf{p}_j}{\|\mathbf{p}_{j-1}-\mathbf{p}_j\|}$ (See Figure 4.2). Starting from the Dirichlet boundary, $j = 0, \cdots, s-1$ is an index of the way-points, s is the number of the way-points on the origin side and $\mathbf{p}_j$ is position of jth way-point. The force at the Dirichlet boundary $\tilde{\mathbf{f}}$ is transmitted to the end of wire, applying tension forces to the way-points:

$$\mathbf{f}_j^- = \|\tilde{\mathbf{f}}\|\mathbf{u}_j, \quad \mathbf{f}_j^+ = -\mathbf{f}_{(j+1)}^-$$

where $\mathbf{f}_j^-$is the tension force of jth way-point, and $\mathbf{f}_j^+$ is a reaction to the force $\mathbf{f}_{(j+1)}^-$, resulting in conservation of total momentum.

$$\mathbf{f}^{\text{origin}} = (\mathbf{f}_0^T, \mathbf{f}_1^T, \cdots, \mathbf{f}_{(s-1)}^T)^T$$

where $\mathbf{f}_j = \mathbf{f}_j^- + \mathbf{f}_j^+$. We apply the same procedure to the way-points on the insertion side and lump the forces into single vector $\mathbf{f}$.

A single muscle volume may contain multiple individually activated motor units. Our model captures this ability by attaching multiple serial elements to individual Dirichlet nodes of a muscle mesh, and modeling several distinct contractile elements within a single simulation volume. For example, the Biceps consist of two motor units (Figure 4.2), one originating from the long head, the other from the short head. Our upper body model incorporates a total of 130 motor units.

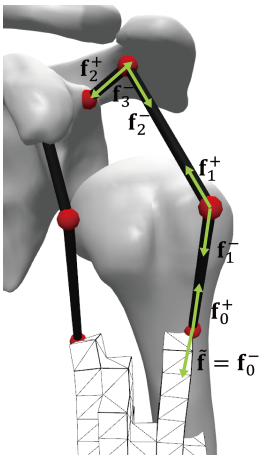


Figure 4.2: Transmission of muscle force to bone. A rigid transformation is applied such that a vector in the direction of the central (contractile) line segment will be rotated parallel to the line segment adjacent to the insertion.

4.2.3 Skeleton model

The Euler-Lagrange equations for the dynamics using generalized coordinates can be represented as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_m^T \mathbf{f}_m(\mathbf{a}) + \mathbf{J}_{\text{ext}}^T \mathbf{f}_{\text{ext}} \tag{4.10}$$

where $\mathbf{q}$ is the vector of joint angles, $\mathbf{M}(\mathbf{q})$ is the generalized mass matrix, $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ represents the Coriolis and gravitational forces, $\mathbf{f}_m = (\mathbf{f}_{(0)}^T, \mathbf{f}_{(1)}^T, \cdots, \mathbf{f}_{(r-1)}^T)^T$ are the muscle forces with the number of muscles r, $\mathbf{f}_{(i)}$ is the force acting on ith muscle and its way-points, $\mathbf{f}_{\text{ext}}$ is the external force, $\mathbf{J}_m$ and $\mathbf{J}_{\text{ext}}$ are Jacobians which map generalized coordinates to Cartesian coordinates, and $\mathbf{a} = (a_0, a_1, ..., a_{r-1})$ are the muscle activation levels, where a_i corresponds to the ith motor unit. Internally, $\mathbf{J}_m$ includes all the information of the muscle attachment points.

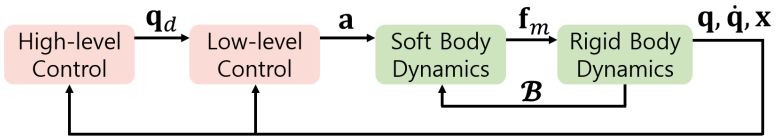


Figure 4.3: The overview of our simulation and control framework: Rigid body states set Dirichlet boundaries for muscle force computation. Our hierarchical controller takes both rigid and soft body states as input and optimizes joint trajectories and muscle activation levels.

4.2.4 Coupling

Forward simulation for the musculoskeletal model proceeds by synchronous evolution of the FEM simulator and the rigid body simulator. At time t_n, for a given skeletal pose the boundary conditions for the volumetric muscle simulation are computed via the Serial Elements, as mentioned earlier in this section. The FEM simulator solves for the equilibrium shape of the muscle volume, in accordance with equation (4.5). Muscle forces are transmitted from the Dirichlet nodes of simulated meshes to the bones via the Serial Element, and finally the dynamic state of the skeleton is advanced using the equations of motion, as summarized in Algorithm 1.

4.3 control

We animate the musculoskeletal model by proposing a two-level hierarchical controller. The low-level controller tracks the desired motion on a per-frame basis and the high-level controller optimizes the motion given a specific task, such as juggling. Combining the robustness of the low-level controller and the generality of the high-level controller, our model can generate a diverse range of desired motion patterns under dynamic situations.

Algorithm 1: Forward Simulation

Data: $\mathbf{a}(t)$: muscle activation levels.

begin

$\mathbf{x}_0$: initial positions of the soft body.

$\mathbf{q}_0, \dot{\mathbf{q}}_0$: initial positions and velocities of rigid body.

for $t = t_0, t_1, \cdots,$ **do**

$\mathcal{B} \longleftarrow$ SetBoundaryConditions($\mathbf{q}_i$)

$\tilde{\mathbf{f}} \longleftarrow$ SolveQuasiStatics($\mathcal{B}, \mathbf{x}_i, \mathbf{a}(t_i)$)

$\mathbf{f}_m \longleftarrow$ TransferForces($\tilde{\mathbf{f}}$)

$\mathbf{q}_{i+1}, \dot{\mathbf{q}}_{i+1} \longleftarrow$ ForwardDynamics($\mathbf{q}_i, \dot{\mathbf{q}}_i, \mathbf{f}_m, \mathbf{f}_{\text{ext}}$)

end

end

The main objective of the low-level controller is to find optimal activation levels for all muscles, tracking a given reference motion at each frame. To achieve this, we adapt the QP-based control method [49, 13] to our model with volumetric muscles. We explain how the muscle Jacobians are computed by leveraging the quasistatic simulation assumption for muscle volumes, and how precomputed factors in the Projective Dynamics formulation can accelerate their computation. Using the low-level control, our high-level controller constructs the reference motion by solving an optimization problem at each frame. Our controller can flexibly generate desired motion patterns using parameterized curves without the need for any motion capture or key framing data.

4.3.1 Low-level controller

The goal of the low-level controller is to compute muscle activations for tracking the desired motion on a per-frame basis. A slight complication is that our

musculoskeletal model is under-determined because the number of degrees of freedom for the skeleton is smaller than that for the muscle activations. Thus, there are many solutions for the same pose, and among them our system must choose one. As detailed below, we compute the solution minimizing three objectives (tracking, effort, and smoothness) through the optimization.

Tracking.

To track the given motion $\mathbf{q}_d$, we compute the desired acceleration $\ddot{\mathbf{q}}_d$ based on PD (Proportional Derivative) control and penalize the difference between the actual joint acceleration $\ddot{\mathbf{q}}$ produced by muscle actuation.

$$\begin{aligned} E_{\text{tracking}} &= w_{\text{tracking}} \|\ddot{\mathbf{q}}_d - \ddot{\mathbf{q}}\|^2, \\ \ddot{\mathbf{q}}_d &= k_p(\mathbf{q}_d - \mathbf{q}) + k_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \end{aligned} \tag{4.11}$$

where k_p, k_v are gains for PD control.

Effort.

Humans move so as to minimize the required effort [104]. Thus, we introduce an objective function that penalizes effort by minimizing the required muscle activations.

$$E_{\text{effort}} = w_{\text{effort}} \|\mathbf{a}\|^2 \tag{4.12}$$

Smoothness.

Physiologically, in activation dynamics, there are processes that convert neural signals to muscle activations [4]. This prevents sudden changes in activation levels. Thus, we penalize the variation of current muscle activations.

$$E_{\text{smooth}} = w_{\text{smooth}} \|\dot{\mathbf{a}}\|^2 \tag{4.13}$$

Using the three objectives described above, our low-level controller can be formulated as the following quadratic program:

$$\begin{aligned} \min_{\ddot{\mathbf{q}},\mathbf{a}} \quad & E_{\text{tracking}} + E_{\text{effort}} + E_{\text{smooth}} \\ \text{subject to} \quad & \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{J}_m^T \mathbf{f}_m(\mathbf{a}) + \mathbf{J}_{\text{ext}}^T \mathbf{f}_{\text{ext}} \\ & 0 \le a_i \le 1 \quad \text{for} \quad i = 0, 1, \cdots r-1 \end{aligned} \tag{4.14}$$

where the equality constraints keep the equations of motion of the skeleton, and the inequality constraints enforce all muscle activations in the range of $[0, 1]$.

It is important to appreciate that equation (4.14) is *nonlinear* on the activations $\{a_i\}$, due to the nonlinearity of the muscle force $\mathbf{f}_m(\mathbf{a})$. The reason for this nonlinearity is subtle, and can be best elucidated by considering the line-segment idealization of the Hill-type primitive (e.g. [49]). In accordance with this model, *for a given muscle length*, the muscle force is an affine function of the activation. However, when the aggregate *length of the musculotendon* is held constant, e.g. for a given skeletal pose, activation of the contractile element will alter the lengths of the muscle and tendon individually, even if their sum is held constant. This variation of the muscle length infuses an additional nonlinearity in the muscle force as a consequence of activation, rendering the muscle force no longer an affine function of activation.

This nonlinearity is often consciously overlooked [49], by making an assumption that the muscle length variation is minimal; there would in fact be no error in this approximation if the tendon was infinitely stiff. With volumetric muscles, however, this approximation would be inexact even if the tendons were fully inextensible, as the volumetric contractile muscle has the potential to alter its geometry via non-uniform deformation that can certainly alter the muscle force produced, even if the longitudinal length of the muscle remained constant. An accurate linearization of the constraint equation (4.14) needs to

employ a proper first-order Taylor exansion $\mathbf{f}_m(\mathbf{a}) \simeq \mathbf{f}_m(\mathbf{a}^*) + \frac{\partial \mathbf{f}_m}{\partial \mathbf{a}}(\mathbf{a} - \mathbf{a}^*)$, for which the Jacobian $\mathbf{J} = \partial \mathbf{f}_m / \partial \mathbf{a}$ needs to be evaluated.

4.3.2 Jacobian Computation

Notably, no closed-form expression of the force $\mathbf{f}_m(\mathbf{a})$ exists, since it incorporates the solution of nonlinear quasistatic equilibrium problem. It is, however, possible to compute its Jacobian *analytically* via careful differentiation of the quasistatic equilibrium condition. For any given value of the Dirichlet conditions applied at the endpoints of the muscle volume, denote by x the *free* nodes of the simulation mesh (excluding Dirichlet boundaries). Let us denote by $\mathbf{x}^*(\mathbf{a})$ the equilibrium positions of these nodes, under applied activations $\mathbf{a}$. Inserting these values in the expression of the total (passive and active) force $\mathbf{f}(\mathbf{x}, \mathbf{a})$ satisfies, by definition, the quasistatic equilibrium condition $\mathbf{f}(\mathbf{x}^*(\mathbf{a}), \mathbf{a}) = 0$. Differentiating with respect to $\mathbf{a}$ yields:

$$\frac{\partial}{\partial \mathbf{a}} \mathbf{f}(\mathbf{x}^*(\mathbf{a}), \mathbf{a}) = \frac{\partial \mathbf{f}}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}^*}{\partial \mathbf{a}} = \mathbf{0} \tag{4.15}$$

where all partial derivatives of forces are evaluated at $\mathbf{x} = \mathbf{x}^*(\mathbf{a})$. This allows us to compute the Jacobian of the muscle force under constant quasistatic equilibrium conditions, as follows:

$$\mathbf{J} = \frac{\partial \mathbf{f}_m(\mathbf{x}^*(\mathbf{a}), \mathbf{a})}{\partial \mathbf{a}} = \frac{\partial \mathbf{f}_m}{\partial \mathbf{a}} + \frac{\partial \mathbf{f}_m}{\partial \mathbf{x}} \frac{\partial \mathbf{x}^*}{\partial \mathbf{a}} \tag{4.16}$$

Closed form expressions of $\partial \mathbf{f}_m / \partial \mathbf{x}$ and $\partial \mathbf{f}_m / \partial \mathbf{a}$ are readily available, while the derivative of the quasistatic solution $\partial \mathbf{x}^* / \partial \mathbf{a}$ is computed by solving the linear system (4.15). Incidentally, the coefficient matrix of this system (i.e. the stiffness matrix $\partial \mathbf{f} / \partial \mathbf{x}$) has already been factorized for the needs of the Projective Dynamics simulation.

4.3.3 High-level Controller

The high-level controller adjusts the reference motion of the upper body model in order to control it in a delicate fashion. We adopted the optimal control technique to solve this finite horizon problem

$$\begin{aligned} \min_{\mathbf{q}_d(t)} \quad & J(\mathbf{s}(t), \mathbf{q}_d(t)) \\ \text{subject to} \quad & \mathbf{q}_{\text{lower}} \leq \mathbf{q}_d(t) \leq \mathbf{q}_{\text{upper}} \\ & \dot{\mathbf{s}}(t) = \mathbf{g}(\mathbf{s}(t), \mathbf{q}_d(t), t) \quad \text{for} \quad 0 \leq t \leq t_f \\ & \text{given} \quad \mathbf{s}(0) = \mathbf{s}_0 \end{aligned} \tag{4.17}$$

where $\mathbf{q}_d(t)$ is the trajectory to be optimized, $\mathbf{s}(t)$ are the states of the system which contain positions and velocities of the rigid bodies, and the positions of all soft bodies, $\mathbf{q}_{\text{lower}}$ and $\mathbf{q}_{\text{upper}}$ are joint limits, $\mathbf{g}(\mathbf{s}(t), \mathbf{q}_d(t), t)$ governs the dynamics of the system, and $J(\mathbf{s}(t), \mathbf{q}_d(t))$ is an objective function which describes the high-level tasks. To convert this infinite-dimensional problem into a finite-dimensional optimization, we parametrize the joint trajectory $\mathbf{q}_d(t)$ with a Cubic Bézier spline as follows:

$$\mathbf{q}_d(t) = \sum_{i=0}^{3} \mathbf{B}_i(t)\mathbf{c}_i \tag{4.18}$$

where $\mathbf{B}_i(t)$ is the basis of Bézier spline, and $\mathbf{c}_i$ are the control points. By parameterizing $\mathbf{q}_d(t)$, the original problem is changed to the finite-dimensional problem of optimizing the control points $\mathbf{c}_i$. Once the optimal $\mathbf{c}_i$ values have been found, the reference trajectory $\mathbf{q}_d, \dot{\mathbf{q}}_d$ is readily computed through the basis functions of the spline. Specifically, in the juggling problem, our optimization

problem is described as follows:

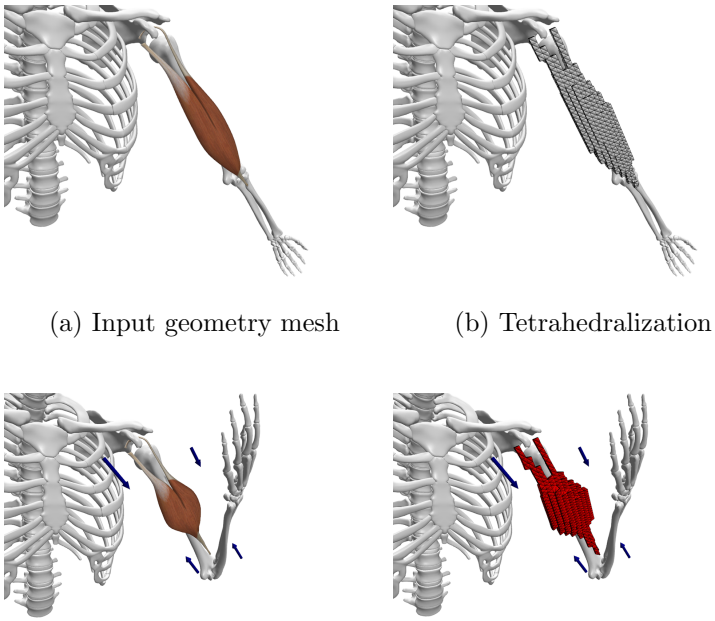
$$\begin{aligned} \min_{\mathbf{c}_0,\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3} \quad & w_p\|\mathbf{p}_{\text{desired}} - \mathbf{p}_{\text{ball}}\|^2 + w_v\|\mathbf{v}_{\text{desired}} - \mathbf{v}_{\text{ball}}\|^2 \\ \text{subject to} \quad & \mathbf{q}_{\text{lower}} \leq \mathbf{c}_i \leq \mathbf{q}_{\text{upper}} \\ & \dot{\mathbf{s}}(t) = \mathbf{g}(\mathbf{s}(t), \mathbf{q}_d(t), t) \quad \text{for} \quad t_0 \leq t \leq t_f \\ & \text{given} \quad \mathbf{s}(t_0) = \mathbf{s}_0 \end{aligned} \tag{4.19}$$

Since the Bézier spline satisfies the convex hull property, just by applying bounds to the control points $\mathbf{c}_i$, we can produce a trajectory $\mathbf{q}_d(t)$ that is bounded above and below. The first term of the objective function penalizes the difference between the ball position and the desired position at the end of the swing phase. The second term penalizes the difference in velocity. A single evaluation of the energy requires simulations through the entire time interval $t_0 \leq t \leq t_f$; thus this is a performance-sensitive optimization operation.

4.4 Experimental results

We implemented our muscle-driven control system in C++. The open source library DART was used for articulated body simulation [97]. Our upperbody musculoskeletal model includes 8 joints: two wrists, two elbows, two shoulders, two collar bones, and one torso. The wrist, the shoulder, and the torso are 3-DOF ball-and-socket joints and all the others are 1-DOF revolute. The model includes 72 muscles that affect the actuation of arm joints.

We used IPOPT [105] to solve our per-frame optimization in equation (4.14). The control optimization and FEM simulation are updated at the rate of 200 Hz, while the articulated body dynamics is integrated at the rate of 1000 Hz. We used the gradient descent method for trajectory optimization with numerical differentiation of the objective function. Trajectory optimization requires 10 to



(a) Input geometry mesh (b) Tetrahedralization

(c) Deformed geometry mesh (d) FEM simulation

Figure 4.4: Muscle modeling and rendering

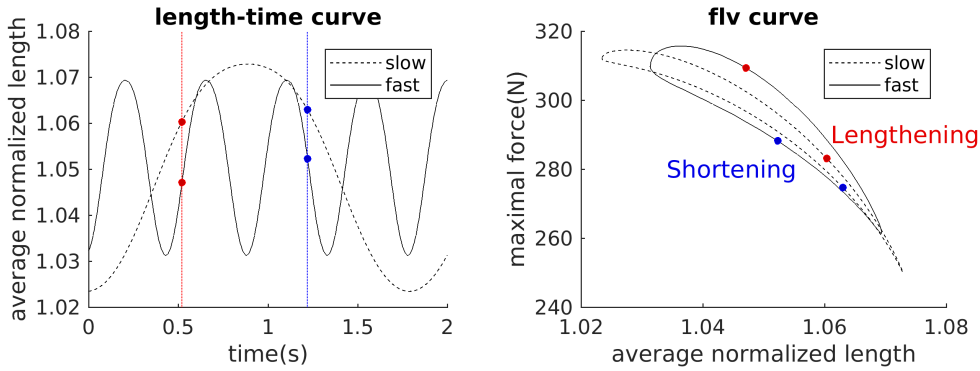


Figure 4.5: Force-length-velocity curves. (Left) Maximal contractile force at the origin of the Biceps short head when it lengthens and shortens periodically at two different speeds. (Right) The Biceps generates larger force when it lengthens quickly.

20 gradient descent iterations to converge and usually takes 10 to 15 minutes per iteration on an Intel i7-6700K 4.0GHz CPU.

4.4.1 Muscle Geometry and Tetrahedralization

Starting with high-resolution geometric meshes of bones and muscles (Figure 4.4(a)), we annotated the origin and insertion of upperbody muscles in 3D geometry and tetrahedralized each individual muscle. The resolution of tetrahedralization is determined so that geometric features, such as bifurcating heads of the Biceps, are expressed clearly (Figure 4.4(b)). In our model, Biceps, Triceps, and Deltoids have about one thousand tetrahedra for each and all the others are simpler. The tetrahedral mesh undergoes deformation via FEM simulation (Figure 4.4(d)). We deform the original geometry mesh accordingly for the visualization of muscle contraction (Figure 4.4(c)). To transfer the deformation of the tetrahedral mesh to the geometric mesh, The 3D vertex location in the geometric mesh is expressed by the barycentric coordinates in its enclosing

tetrahedra. Most of the vertices are enclosed by the tetrahedra and only small outliers fall outside the tetrahedral mesh. Each outlier vertex is mapped to the closest tetrahedron and its barycentric coordinates would have negative values.

Although high-resolution tetrahedral meshes are ideal for the accuracy of FEM simulation, low-resolution, hand-crafted meshes are also useful for the efficiency of simulating dexterous tasks. We used high-resolution meshes to examine the functionality and strength of each individual muscle under various conditions, and low-resolution meshes to simulate and control two-hand manipulation tasks.

The volumetric muscles driven by FEM simulation inherit the contraction dynamics of the Hill-type model. Each volumetric muscle generates its maximal contractile force when it is at its rest length and becomes weaker when it lengthens or shortens. The elasticity of the deformable material prevents it from lengthening excessively. The force-length-velocity curve in Figure 4.5 shows velocity-dependent contraction dynamics. The muscle generates its maximal force when it lengthens, which is called *eccentric contraction*.

4.4.2 Juggling

Juggling is a sophisticated performance with two hands for entertainment, art, or sports. A juggler usually manipulates more than three balls at the same time, while all of them are dynamic. The balls are repeatedly thrown by one hand, floating in the air, and finally captured by the other hand.

Mathematical expressions for juggling can be explained by the Siteswap value. Let T be the time per beat, D be the ratio of holding time that the ball spends in the hand per beat, and V be the Siteswap value which defines the pattern of juggling [106]. For each beat, the ball should be thrown such that it lands on the other hand after V beats. For example, if the sequence of V is

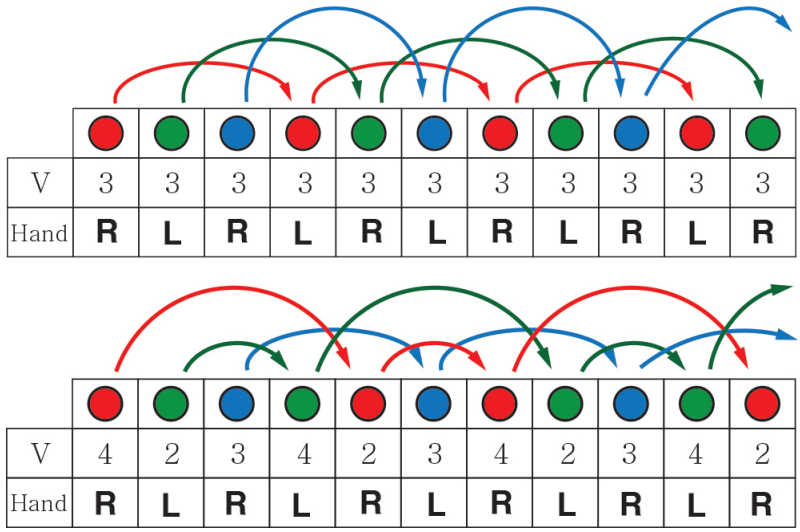


Figure 4.6: Siteswap patterns of (up) 333 and (down) 423 juggling.

$3, 3, 3, \cdots$, the first ball thrown by the left hand will land on the right hand after 3 beats. While the first ball is still in the air, the right hand pitches the second ball after one beat. Figure 4.6 illustrates Siteswap patterns.

The maximum height of the ball is proportional to its Siteswap value V. Provided that T, D, and the sequence of V are given, we can determine the time of flight of individual balls, $t_{\text{flight}} = T(V - 2D)$. Note that we multiply D by two because one round trip of the ball takes two swing phases. If the motion of the ball is parabolic in the Y-axis, we can determine the initial velocity $\mathbf{v}_y = \frac{1}{2} g t_{\text{flight}}$, where $g = 9.8 m/s^2$ is the gravitational acceleration.

Our controller generates hand trajectories based on a finite state machine, where each state specifies either swing or catch tasks (Figure 4.7). For each beat, catch action moves the hand toward the landing position by solving inverse kinematics of the hand. Once the ball lands into the hand, a zero-DOF, welded joint is used to attach the ball to the hand. Swing (pitch) action requires

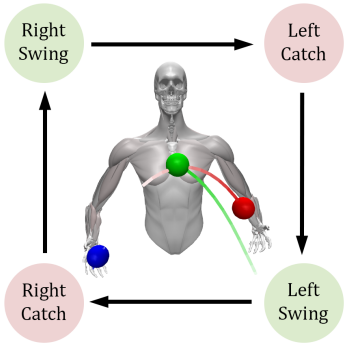
trajectory optimization to match the desired position and velocity at the end of the swing phase.

Juggling Patterns

A *Cascade* is the simplest juggling pattern that pitches the balls to the same height. There are many juggling patterns other than Cascade patterns, varying the height of the balls, the symmetry/asymmetry of patterns, the shape of the projectiles, and the number of jugglers. We first demonstrate cascade patterns starting from 3 balls and adding balls one-by-one to end up with 5 balls (Figure 4.8(a)). With more balls, the juggler has to pitch them higher to maintain the cascade pattern. Our control system can seamlessly adapt to the addition/removal of balls and the switching between juggling patterns. Our muscle-driven control system can also simulate non-cascade juggling patterns. 423 juggling is a non-cascade pattern using three balls. The juggler pitches two balls higher than the third (Figure 4.8(b)). 64 juggling exhibits an asymmetric pattern with five balls. The right hand juggles with three balls, while the left hand independently juggles with the other two (Figure 4.8(c)). Our controller adaptively optimizes the swing trajectory to pitch the balls toward the desired direction at the desired speed, starting from the same initial configuration and parameter settings.

External Perturbation

We tested our controller under external pushes. Random forces of magnitude 300N and duration 0.5s are exerted on the torso, shoulder and elbow joints. Our solver re-optimizes the disturbed swing trajectory to adapt to the pushes. Using the original trajectory as initial guess, re-optimizing the trajectory takes only a few iterations to converge. The controller can endure large pushes on the



k_p	500	w_{tracking}	1.0
k_v	44.7	w_{effort}	10.0
		w_{smooth}	10.0
w_p	0.01	T	0.3s
w_v	0.01	D	0.7
$t_f = T \cdot D$	0.21s	V	$2to6$

Figure 4.7: A finite state machine for juggling and simulation parameters.

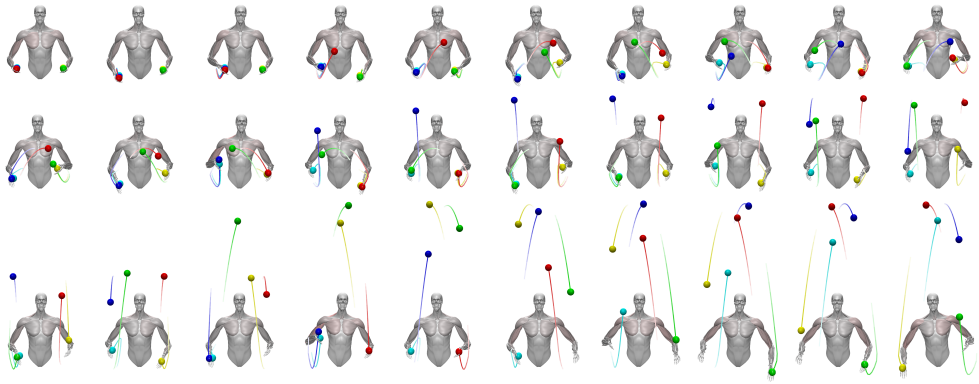
torso and the shoulder, while even small perturbations at the extremities could be critical for dexterous manipulation such as juggling.

Mass Variations

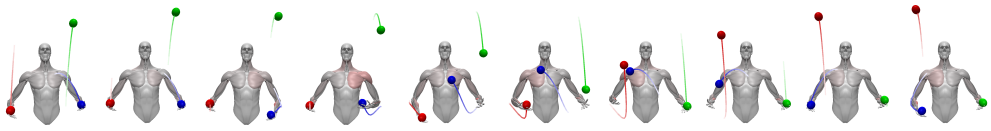
This example shows the Cascade pattern with three balls of different mass (0.1kg, 0.5kg and 2kg). All three are thrown to the same height regardless of their mass difference and consequently their flight duration is the same. Having the flight time fixed, the response to different mass necessarily leads to the modulation of the swing duration. We regulate the release timing t_f of the ball in response to the loading mass.

$$t_f = c_1 m_{\text{ball}} + c_2 \tag{4.20}$$

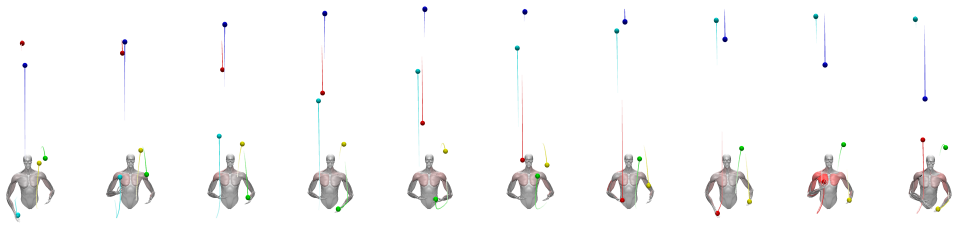
where c_1 and c_2 are scalar coefficients and m_{ball} is the mass of the ball. The hand pulls back further with the heavy ball to absorb impact and travels longer to compensate for the mass. Our controller also accounts for the physiological property of the muscles, which can generate larger force when they are in eccentric contraction. Eccentric contraction of major agonistic muscles occurs when the swing arm pulls backward.



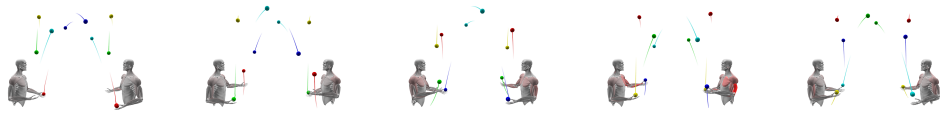
(a) 345 juggling



(b) 423 juggling



(c) 64 juggling



(d) Two person juggling

Figure 4.8: Juggling patterns.

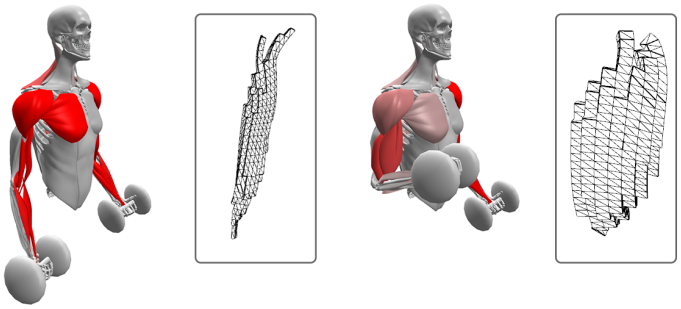


Figure 4.9: Atrophy and Hypertropy of Biceps and Brachialis in the right arm. (Left) The crossectional area is scaled down by a factor of 0.5. (Right) The crossectional area is scaled up by a factor of 1.5. Muscle hypertrophy allows the weight (10 kg) to be lifted easily at low muscle activation levels.

4.4.3 Muscle Disorder

There are many types of muscle diseases with different causes and outcome. With muscle models simplified to line segments, we do not have many options to formulate the symptoms of muscular disorders into computational models. The most popular approach is to manipulate the force-length and force-velocity curves of the Hill-type model, which governs the muscle contraction dynamics. The Hill-type model is an analytic function based on in vitro measurement of muscle deformation and material properties. Therefore, manipulating the Hill-type model is an indirect approach based on approximations. The use of volumetric muscles opens up new possibilities in this regard, since the geometry and material properties of volumetric cells can be specified and manipulated directly.

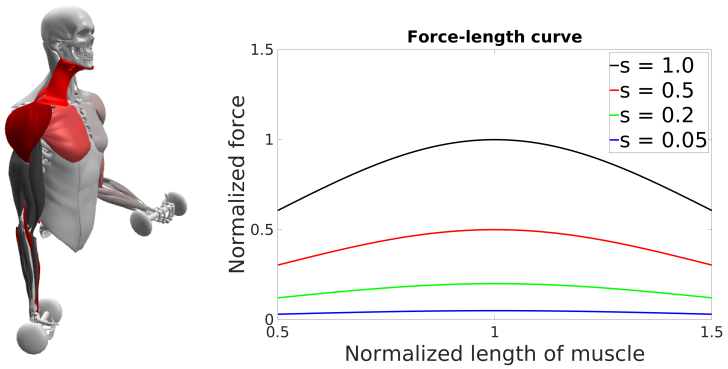


Figure 4.10: Muscle weakness simulation. The weight of the dumbbell is 5kg. (Left) The muscles in the right arm shown in dark brown are weaker than normal ability. (Right) The scaling of the force-length curve determines the level of weakness.

Atrophy and Hypertrophy

Atrophy indicates the loss of mass and strength of muscles, which can cause disability or difficulty of actions. Conversely, *hypertrophy* is the increase of muscle volume and enhanced muscle strength. The symptoms of atrophy and hypertrophy can be simulated by changing the geometry of the tetrahedral mesh. As suggested by Kadlecek et al. [26], we scaled the cross-section of the tetrahedral mesh by a factor of 0.5 (atrophy) and 1.5 (hypertrophy) to observe the weakening and strengthening of muscle capacity (Figure 4.9).

Deficiency and Paralysis

Since the material property of our volumetric muscles is derived from the Hill-type model, we can use the curve trick to simulate muscle deficiencies (Figure 4.10). The force-length curve indicates the strength of the muscle. In our simulation, scaling the magnitude of the force-length graph by a factor of 0.5,

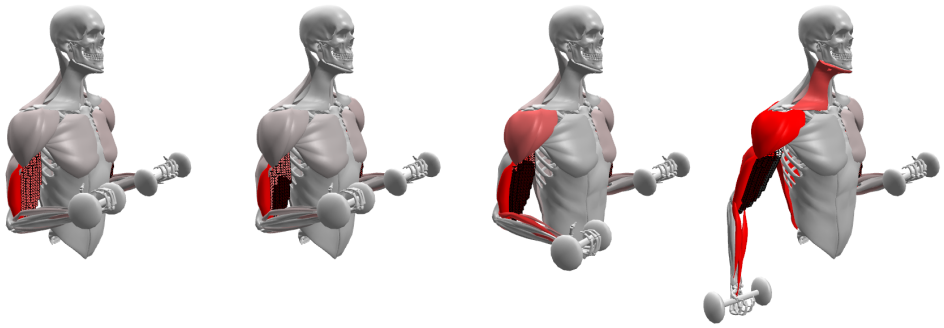


Figure 4.11: Progressive paralysis of Biceps and Brachialis in the right arm. (From left to right) As dark cells spread, Biceps and Brachialis become weaker and therefore the nearby muscles in the forearm and the shoulder activate more to compensate for the weakness.

0.2, and 0.05 resulted in the progressive weakness of the muscle. We can observe the increased activation of the nearby muscles which compensate for the weakness. The effect of such graph scaling always affects the entire muscle if the model is simplified to a line segment. Our volumetric model offers the flexibility to edit the material property of individual cells. Figure 4.11 demonstrates the effect of paralysis spreading progressively over the Biceps and Brachialis. The deactivated cells shown in dark brown do not generate any contractile force since the muscle-length curve is set to zero at all lengths.

Contracture

Contracture is the shortening or stiffening of muscles, that results in decreased movements and range of motion (Figure 4.12(middle)). The symptoms of contracture can be simulated by manipulating either the force-length curve or the volumetric mesh. With the Hill-type model, muscle shortening is described by its passive element that engages earlier than the normal force-length curve.

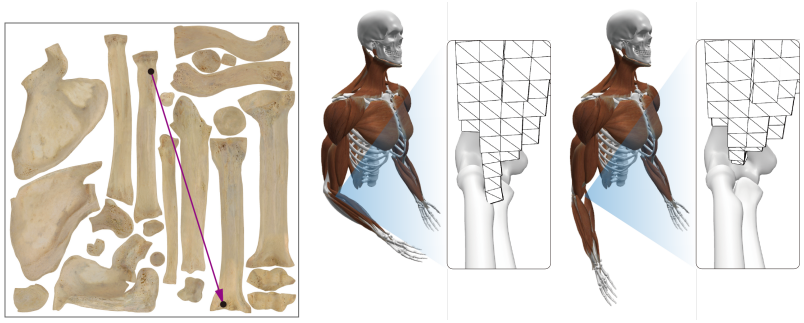


Figure 4.12: Muscle contracture and orthopedic surgery simulation. (Left) A 2D point on the texture atlas maps to a 3D point on the bones. The user can easily specify the new insertion of the muscle on the texture atlas. (Middle) The increased tension of the contracted Biceps results in the flexed elbow at the relaxed arm. (Right) The surgery simulation displaces the insertion of the Biceps from the top of the *Radius* to the bottom of the *Humerus*. As a result, the Biceps becomes a one-joint muscle. Since the surgery increases the range of motion, the elbow becomes fully extended at the relaxed arm. The simulation also confirms the side-effect that the maximum torque at the elbow becomes weaker with the displaced muscle insertion.

Alternatively, our volumetric muscles offer a simpler, more intuitive approach. Shrinking the rest shape of the tetrahedral mesh results in increased tension between the origin and insertion of the contracted muscle.

Orthopedic Surgery Simulation

Contracture occurs frequently in patients with cerebral palsy. The treatments include orthopedic surgery, which either lengthens musculotendons or reduces the tension by displacing the insertion of the muscle. A muscle that is a flexor at one joint may also be a flexor or extensor at another. These are called *two-joint muscles*, which span across two joints. Displacing the insertion across a joint by surgery can turn a two-joint muscle into one-joint. This type of surgery not only reduces muscle tension, but has the side-effect of changing the functionality of the muscle. Therefore, being able to evaluate the effects and side-effects of the surgery is of practical importance. We implemented a simple user interface system to simulate orthopedic surgery (Figure 4.12). The user interface shows an atlas of bone texture maps and allows the user to specify a new insertion point on the atlas. Our simulation system updates the muscle geometry instantly and reflects the update to the musculoskeletal simulation, visualizing the effects of the surgery.

4.4.4 Limitations and Failure Cases

Even though successful applications of volumetric muscles have been demonstrated so far, we have also faced failure cases. First, the shoulder range of motion of our model is narrower than normal, making it difficult to raise the arm over the head. The range of joint motion is influenced by many factors including the geometric configuration of muscle origins, insertions, and their way-points, muscle strength, the discretization of simulation meshes, and the

choice of a contraction dynamics model. Furthermore, parameters of Hill-type curves affect the stability of dynamics simulation, as discussed by Sachdeva et al. [18]. The negative slope of the force-length curve could incur numerical instability and consequently restrict the range of motion. We found it non-trivial to hand-tune parame.ters to achieve a desired range of motion. Optimization-based automatic parameter tuning is highly desirable in future study.

Secondly, the generated motion looks stiffer than we expected. The main cause of stiffness is large PD gains, which are necessary for accurate control. Juggling in particular requires precise aiming and timing of the balls. There is a trade-off between control accuracy and motion stiffness. Even though the total sum of muscle activation is minimized during trajectory optimization, the influence of large PD gains still remains to a certain extent. A potential remedy is the use of variable PD gains over the trajectory. Large gains are necessary only when it throws a ball with precision aiming at the end of arm swing. Small gains are preferred in the middle of arm swing for motion compliance. Variable PD gains optimized together with arm trajectory would alleviate stiffness without sacrificing accuracy.

4.5 Discussion

Our attempt to transition the composition of an active simulated musculoskeletal system from the established practice of line-segment approximations to volumetric primitives represents a significant step towards the ultimate goal of a true biomemetic digital replica of the human anatomy and its complexity. Nevertheless, our current framework still consciously submits to a number of limitations, both in its scope of applicability, as well as biomechanical accuracy and computational capability. A number of these limitations stem from the

computational cost of incorporating a system with a large number of dynamic degrees of freedom inside a full control loop; we expect that the continually increasing computational capabilities of current platforms provide an encouraging roadmap for addressing such performance-oriented challenges. Secondly, the degree of biomechanical accuracy that our system can afford is restricted by the limited availability of highly detailed digital models of anatomy, not only in terms of geometrical shapes, but also in terms of the governing laws (material constitutive models, fiber fields of anisotropic contractile muscles, mechanical response models of fascia and viscoelastic tissues). We aspire that the foundation we have laid will allow us to leverage better models and governing laws, as those crystallize and become validated in relevant literature.

A number of technical considerations that complicate our pursuit of biomechanical accuracy result from the fact that the very nature of the line-segment simplification conceals certain challenges that are inherently present in real volumetric musculature. A line segment muscle primitive is constrained, by definition, to retain the shape of a straight line between any two successive via-points (or in its entirety, if the muscle primitive is not segmented). As a consequence, when the skeleton is articulated in a way that would cause the aggregate length of the musculotendon to shrink from its rest length, there is no ambiguity as to what the resulting shape of the primitive would be: it remains a piecewise linear curve. If a true volumetric muscle was modeled in complete isolation from its surrounding passive/connecting tissue and adjoining muscles (in direct analogy to how line-segment primitives are), there would be ambiguity in its resulting equilibrium shape, as there is a multitude of directions in which the geometry of the muscle could be laterally deflected, buckled or bent (in the case of tendons). This incurs a degree of ambiguity in the muscle forces that a compressed volumetric muscle produces as a result of skeleton-induced

reduction of the length of the medial axis of the musculotendon (this is alleviated, in part, in scenarios where muscle activation incurs tension in the muscle). This behavior is stabilized, in reality, by the contact and collision between the muscle and its surrounding tissue, which helps resolve the resulting equilibrium shape. However, in our initial exploration presented in this work, we have not incorporated explicit contact and collision handling between muscle volumes and their surrounding tissues. As a consequence, our simulated muscles may experience bending or buckling modes under compression that are not fully representative of the biophysical behavior. From the standpoint of numerics, this also induces a practical limitation in the minimum thickness we can allow the tendon regions to assume, as excessively thin tendons would both increase this modeling error, and aggrevate the presence of inaccurate buckling modes. We expect this deficiency to be cured in future work by the incorporation of careful contact and collision processing between a fully-coupled set of muscles and passive/connective tissue.

In the real human body, individual muscle volumes are mechanically correlated via their contact coupling. For example, it is possible for the insertions of a given muscle to exert tension to the skeleton even if the muscle is fully inactive, at a kinematic state where the line-segment approximation would yield no tension at all; this would be possible if the contraction of a neighboring muscle, coupled via connective tissue and contact handling, causes a volumetric deformation in the inactive muscle to be deformed as a side-effect, producing tension (and forces at the insertions) that would not be possible with fully-independent line-segment muscle primitives. We must highlight that our current formulation only partially captures this real-world behavior, on volumetric muscle primitives with multiple independently-activated contractile regions modeled in the same volumetric simulation mesh; on the other hand, we do not currently capture

this coupling by means of contact processing between distinct muscle volumes, which are simulated independently. Our mathematical formulation for computing force Jacobians is able to capture this dependency on muscles with several contractile regions. Extending this capability (and the Jacobian computation) to contact-coupled muscles will be more challenging, and possibly restrict our options for contact handling (e.g. creating a preference for coupling via "penalty" contact forces, instead of impulse-based corrections mediating momentum exchange [107]).

Finally, the simplicity of the line-segment muscle primitives reduces the complexity of modeling individual muscles to a specification of the muscle path, the muscle/tendon ratio (without any need for localization), and parameters governing its maximum force-generating potential. Volumetric muscle primitives have much broader modeling flexibility, and as a consequence many more opportunities for less-than-accurate geometry or material parameter specifications to create deviations from the ground truth. In a sense, the increased flexibility of volumetric models to approach reality comes at the cost of increased opportunities for modeling errors to create deviations from it. As an example, the cross-sectional geometry of tendons can have a significant effect on the resulting geometry of the muscle volume (if the Young's modulus is set to a near-constant value, as biophysically expected). Line-segment models can mask such parameter tuning choices behind an individualized setting of the tendon stiffness on a muscle-by-muscle basis, making the parameter space easier to tune (even if such parameters have a very loose connection to the underlying first principles). Our existing model consciously focuses on upper body motion; future work should explore extensions to a full-body human model, capable of resolving more diverse motion.

Chapter 5

Deep Compliant Control

In addition to the simulation framework of the human motor system, we consider interactivity of the physically simulated humanoid. When manipulating an object or trying to avoid large impacts, humans consciously plan the sequence of interactions so that the humans utilize interaction forces. This interactivity is crucial even in a static scene where there are no external perturbations, and a human continuously interacts with the ground and takes advantages of ground reaction forces to maintain its balance. Since human musculoskeletal system itself lacks of explicit sensory modules that recognize its surroundings, we can't guarantee the dynamics of interactions. Tactile sensors provide the primary recognition of the interactions, and thus in this chapter we model the tactile sensory system and attach to our learning framework for the motor control. Technical challenges include defining the ways of interactions, reproducing physically reliable movements, and robustly controlling under-actuated dynamical systems. The key technical contribution is a two-level control architecture based on deep reinforcement learning that imitate human movements while

adjusting their bodies to external perturbations. The controller minimizes the interaction forces and the control torques for imitation, and we demonstrate the effectiveness of the controller with various motor skills including opening doors, balancing a ball, and running hand in hand.

5.1 Overview

Humans interact with their surroundings every day. Opening doors, pushing boxes, and playing sports are examples of these interactions. Forces serve as a medium for such interactions, as humans interact with the environment by repeating a dynamic process of sensing these forces and actuating their bodies. In this process, compliance is one of the primary principles which determines human movements during interactions. Compliance provides humans the ability to move in various ways to avoid large amounts of contact forces and adapt to unexpected situations.

This paper aims to build a framework for the simulation and control of humanoids, allowing the humanoid character to interact with their surroundings in physically reliable ways. We can generate a broad spectrum of movements ranging from passive reactions to external physical perturbations, to active manipulations with clear intentions. Technical challenges include defining and modeling of compliance, reproducing realistic human movements, and providing robust control of motor skills for under-actuated dynamical systems. We define the character's compliance from a mechanical point of view. Our motion controller deforms the character to increase compliance. We also present a learning framework based on Deep Reinforcement Learning (DRL).

Recent progress in DRL has shown its robustness in learning motor skills for movements of high-dimensional characters such as humanoids in physics

environments [2, 108, 109, 91]. A control policy (a.k.a controller) learned from DRL allows the character to generate physically plausible motions. The core idea of these methods stems from *imitation learning*. The reference motion data provides the character on how to move, and the controller actuates its character bodies based on their physical states to track the reference motion data in the given environment. While the existence of the reference motion data enables the imitation of human tasks, it hinders the simulated character from learning diverse trajectories, generating stiff motions even when large external perturbative forces are being applied. We present a new two-level hierarchical control algorithm based on DRL which allows the character to flexibly interact and move with its surroundings. The compliant controller is a kinematic layer that adjusts the character pose to external perturbations at high frame rates, while the imitation controller learns the kinematics and the dynamics of the character at low frame rates. Our two-level controller minimizes the interaction forces and the control torques for the imitations. We demonstrate the effectiveness of the controller with various examples:

- The character can stand, walk and run even when the interaction forces dominantly affect the character. Based on the forces, our controller generates heterogeneous movements where the lower extremities balance the full body while the upper extremities deform in accordance with the forces.

- We can predict how the mass distributions affect full body movements. Upon Newton's third law of motion, two characters behave differently in order to gain each individual compliance.

- Our controller can perform manipulation tasks such as opening doors or balancing a ball on the head. We demonstrate our controller's robustness

and capability by adding perturbations to the environment.

5.2 Environment

Our framework assumes the character as articulated rigid bodies, which consists of links and connecting joints. We assume the character is an open-chain (a.k.a tree-structured) articulation with the floating root. The dynamic states of the character can be expressed by its joint positions $\mathbf{q} \in \mathbb{R}^n$ and joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$ in generalized coordinates. The equations of motion describe the dynamical system as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = \tau + \tau_{\text{ext}} \tag{5.1}$$

where $\mathbf{M}(\mathbf{q})$ is the mass matrix and $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis and gravitational forces. τ_{ext} is the sum of external forces including contact forces and other external perturbations. τ is the internal force which is computed from PD servos for each joint. Given target joint positions $\bar{\mathbf{q}}$, PD servos use proportional and derivative errors to track the target positions:

$$\tau = k_{\text{p}}(\bar{\mathbf{q}} - \mathbf{q}) - k_{\text{v}}\dot{\mathbf{q}} \tag{5.2}$$

where k_{p} and k_{v} are gains. At each time step, the system evolves in time by updating the joint positions and velocities via semi-implicit Euler integration.

5.2.1 Stiffness

In classical mechanics, *stiffness* is the measurement of how much an object resists deformation in response to an applied force. The mathematical expression of a 1-dof (degrees of freedom) system is defined as $k = -f/\delta x$, where f is the restoring force, and δx is the displacement. *Compliance* is defined as the inverse of the stiffness $c = k^{-1}$. We can generalize these definitions to high-dimensional

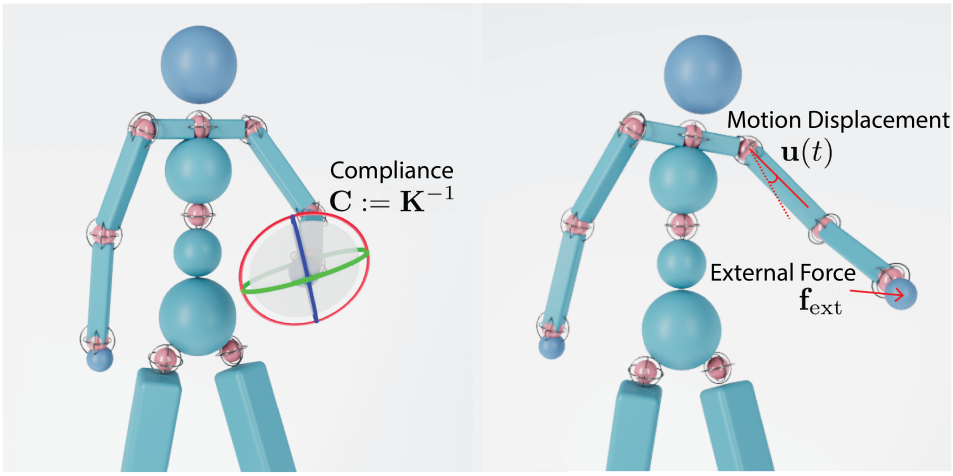


Figure 5.1: Compliance of the left hand. Due to the geometric locking, the character acts stiffly when applied force is directing to the arm while the other directions have moderate compliance.

characters. Unlike the 1-dof spring, stiffness changes dynamically depending on the pose of the character as well as the acting point of an external force. Given a displacement $\mathbf{x} \in \mathbb{R}^3$ produced by the external force, the stiffness of the character is given by:

$$\mathbf{K} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \tag{5.3}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is stiffness matrix and $\mathbf{f} \in \mathbb{R}^3$ is a force that the character actuates due to the displacement $\mathbf{x}$. We first examine the matrix assuming stationary target positions (i.e. $\bar{\mathbf{q}}$ is constant). As for the PD controlled character, we can explicitly compute Equation (5.3) by differentiating Equation (5.2):

$$\mathbf{K} = k_{\text{p}}\mathbf{L} \tag{5.4}$$

where $\mathbf{L} = (\mathbf{J}\mathbf{J}^{\text{T}})^{-1}$ with the Jacobian matrix $\mathbf{J} = \partial\mathbf{x}/\partial\mathbf{q}$ mapping velocities in joint space to velocities in work space defined by the acting point of the external

force. The matrix $\mathbf{L}$ is the inverse of the Laplacian matrix of the articulation graph of the character weighted by moment arms of each body. We detail the derivation of the matrix $\mathbf{K}$ in Appendix. Once the external force is applied at a certain point, the mechanical *stiffness* is readily computed through the Jacobian matrix. We can interpret the matrix by the singular value decomposition (SVD):

$$\mathbf{K} = k_{\mathrm{p}}\mathbf{R}\mathbf{D}\mathbf{R}^{\mathrm{T}} \tag{5.5}$$

where $\mathbf{L} = \mathbf{R}\mathbf{D}\mathbf{R}^{\mathrm{T}}$, $\mathbf{R}$ is a rotation matrix, and $\mathbf{D} = \mathrm{diag}(d_0, d_1, d_2)$ is a diagonal matrix consisting of singular values. Since the matrix $\mathbf{K}$ is a symmetric 3-by-3 matrix, SVD is identical to eigenvalue decomposition. Each singular value indicates an anisotropic stiffness value corresponding to each column of the matrix $\mathbf{R}$, forming an infinitesimal ellipsoid (see Figure 5.1). The stiffness $k = \det(\mathbf{K})$ is computed by taking the determinant of the matrix.

In practice, the stiffness matrix $\mathbf{K}$ depends on the joint target positions. We are able to regulate the stiffness to our preference by modulating $\bar{\mathbf{q}}$. In the following section, we present a novel motion controller, capable of reducing the stiffness of the character.

5.2.2 Compliance-induced Motion Controller

The goal of our controller is to generate joint positions $\bar{\mathbf{q}}(t)$ at every time step to react to external perturbations. The controller takes as input the external forces, the character pose, and the level of compliance to deform the motions. We begin by deriving the stiffness matrix given joint target positions:

$$\hat{\mathbf{K}} := k_{\mathrm{p}}[\mathbf{L} - \mathbf{J}^{-\mathrm{T}}\frac{\partial \bar{\mathbf{q}}}{\partial \mathbf{x}}] \tag{5.6}$$

where $\mathbf{J}^{-1}$ is the pseudo-inverse of the Jacobian matrix. Equation (5.6) indicates the relation between the stiffness and the joint target positions. It is obvious

Algorithm 2: Compliant-induced Motion Controller

Data: $\mathbf{q}^m(t) = (\mathbf{q}_1^m, \mathbf{q}_2^m, \mathbf{q}_3^m, \cdots)$

Result: $\hat{\mathbf{q}}(t)$

$\mathbf{u}_0 \leftarrow \mathbf{0}$

for $i = 1, 2, 3, \cdots$ **do**

$\delta\ddot{\mathbf{q}}_i \leftarrow \text{ForwardDynamics}(\mathbf{f}_{\text{ext}}^i)$

$\dot{\mathbf{x}}_i = h\mathbf{J}\delta\ddot{\mathbf{q}}_i$

$\dot{\mathbf{u}}_i = \alpha\mathbf{J}^{\mathrm{T}}\mathbf{L}\dot{\mathbf{x}}_i$

$\mathbf{u}_i = \mathbf{u}_{i-1} \oplus h\dot{\mathbf{u}}_i$

$\hat{\mathbf{q}}_i = \mathbf{q}_i^m \oplus \mathbf{u}_i$

end

that the stiffness changes when $\bar{\mathbf{q}}$ is a function of the displacement $\mathbf{x}$. If we generate $\bar{\mathbf{q}}(t)$ such that it meets the condition $\partial\bar{\mathbf{q}}/\partial\mathbf{x} = \mathbf{J}^{\mathrm{T}}\mathbf{L}$, the right hand side of Equation (5.6) sums to zero, implying that the character is infinitely flexible. Since this means we lose control of the character entirely, we instead take a middle ground. We modify the condition as follows:

$$\frac{\partial\bar{\mathbf{q}}}{\partial\mathbf{x}} = \alpha\mathbf{J}^{\mathrm{T}}\mathbf{L} \tag{5.7}$$

where $0 \leq \alpha \leq 1$ is the level of compliance. Incorporating Equation (5.7) into the Equation (5.6) results in:

$$\hat{\mathbf{K}} = k_{\mathrm{p}}[\mathbf{L} - \mathbf{J}^{-\mathrm{T}}(\alpha\mathbf{J}^{\mathrm{T}}\mathbf{L})] = k_{\mathrm{p}}(1-\alpha)\mathbf{L} \tag{5.8}$$

which decreases the stiffness $\hat{k} = \det(\hat{\mathbf{K}}) = (1-\alpha)^3 \det(\mathbf{K})$ by the factor of $(1-\alpha)^3$. In our experiments, we choose the level of compliance $\alpha = [0.04, 0.16]$ depending on the magnitude of the forces. It is important to note that $\hat{\mathbf{K}}$ preserves the features of $\mathbf{K}$. Indeed, there exist numerous choices for $\partial\bar{\mathbf{q}}/\partial\mathbf{x}$ that reduce stiffness. The rotation matrix $\mathbf{R}$ is an intrinsic property of the

character that reflects the kinematics of the character. We choose this equation since it conserves $\mathbf{R}$ while decreasing the singular values of $\mathbf{K}$.

Equation (5.7) serves as a first-order dynamics for the motion controller. We define the displacement in generalized coordinates $\mathbf{u}(t) := \delta\bar{\mathbf{q}}(t)$ to produce joint target positions by operating $\hat{\mathbf{q}}(t) = \mathbf{q}^m(t) \oplus \mathbf{u}(t)$, where $\mathbf{q}^m(t)$ is the reference motion. We use the symbol $\oplus$ to denote quaternion multiplications for each joint [96]. The dynamic equation for $\mathbf{u}(t)$ can be written as:

$$\begin{aligned} \dot{\mathbf{u}}(t) &= \alpha \mathbf{J}^{\mathrm{T}} \mathbf{L} \dot{\mathbf{x}}(t) \\ \hat{\mathbf{q}}(t) &= \mathbf{q}^m(t) \oplus \mathbf{u}(t). \end{aligned} \tag{5.9}$$

To compute $\dot{\mathbf{x}}(t)$, we perform forward dynamics of the character to compute the changes of joint accelerations $\delta\ddot{\mathbf{q}}$. Sequentially integrating with time step h and transforming to the work space results in $\dot{\mathbf{x}} = h\mathbf{J}\delta\ddot{\mathbf{q}}$. We update $\mathbf{u}(t)$ by semi-implicit Euler integration. Algorithm 2 shows the overall sequence of the motion controller.

Since the character is under-actuated, we carefully incorporate the root joint into the motion controller. We extract the positional displacement of the root joint $\mathbf{u}_p(t)$ from $\mathbf{u}(t)$ and assume an imaginary linear PD servo for the joint to measure stiffness and the following displacement. Since there is no actual servo at the root joint, we do not actuate the root joint directly. It is required for the character to learn biped locomotion to plan maneuvers driven by $\mathbf{u}_p(t)$. In the following section, we present a framework for learning the motor skills for the physically simulated character equipped with our compliance-induced motion controller.

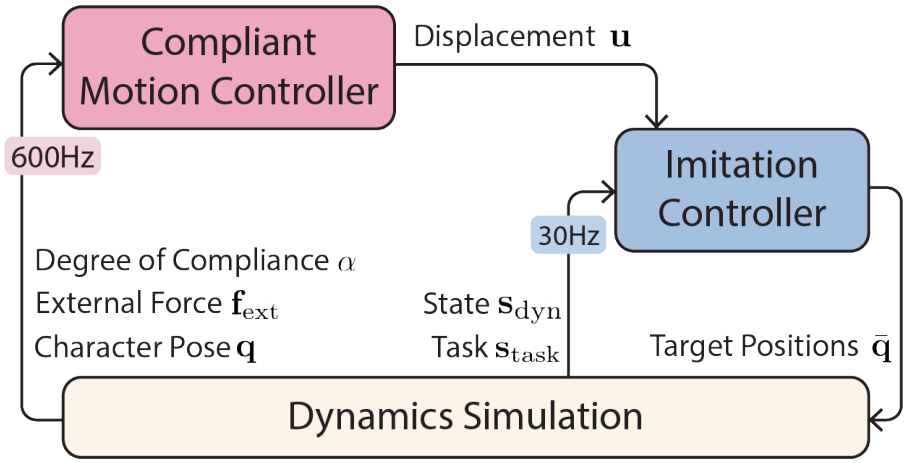


Figure 5.2: System overview.

5.3 Learning Motor Skills

Notably, PD control with the joint target positions $\bar{\mathbf{q}}(t) = \hat{\mathbf{q}}(t)$ will not work; not only is the character an under-actuated system because of the floating base, but it is also affected by various contact constraints and external forces. To remedy this, a more sophisticated feedback controller is required to modulate the joint target positions in accordance with the humanoid state (see Figure 5.2). We use DRL with imitation rewards to learn this feedback controller. The controller $\pi_\theta(\mathbf{a}|\mathbf{s})$ is defined by a deep neural network whose parameters are given by θ to produce an action $\mathbf{a}$. Here, the action is specified by joint target positions for the PD control. Note that we do not actuate the joint by using $\hat{\mathbf{q}}(t)$ directly, but they serve as imitation rewards in order for the character to track $\hat{\mathbf{q}}(t)$ as close as possible. The state $\mathbf{s} = (\mathbf{s}_{\text{dyn}}, \mathbf{u}, \mathbf{s}_{\text{task}})$ includes the dynamic state of the character, the motion displacement of the motion controller, and optionally the state of the tasks. The action is the modulated target positions $\bar{\mathbf{q}}(t)$. In DRL, the controller observes the state $\mathbf{s}_t$ to produce the action $\mathbf{a}_t$ at each time

step. The action is then applied to the system, evolving to the next time step $t+1$. This results in the new state $\mathbf{s}_{t+1}$ and the scalar reward $r_t = R(\mathbf{s}_t, \mathbf{s}_{t+1})$. The reward function encourages the character to mimic the motion, and the goal of the controller is to optimize its parameters to maximize the expected accumulated reward:

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \cdots} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{5.10}$$

where γ is the discount factor, $\mathbf{s}_0 \sim \rho_0$ is sampled from an initial state distribution ρ_0, $\mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t)$ is the action from the controller, and $\mathbf{s}_{t+1} \sim p(\mathbf{s}_t, \mathbf{a}_t)$ is sampled from transition probability p, which specifies the dynamics of the environment. Since evaluation of the objectives requires an intractable computational cost, DRL also learns another network called the value function $V^\pi(\mathbf{s})$ [110, 111]. During each epoch of learning, the controller $\pi_\theta(\mathbf{a}|\mathbf{s})$ and the value function $V^\pi(\mathbf{s})$ are jointly optimized using transition tuples $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$. We use Proximal Policy Optimization, which uses the surrogate loss of the objective to improve the training robustness by clipping the probability ratio of the action [95].

5.3.1 Imitation Reward with Adversarial Network

The beauty of DRL is the ability to learn complex motor skills with the scalar reward function. The reward function r encourages the character to imitate the motion, and optionally guides the character to achieve various task objectives. The reward is designed as:

$$r = w_{\text{imit}} r_{\text{imit}} + w_{\text{task}} r_{\text{task}} \tag{5.11}$$

where r_{imit} is an imitation reward which evaluates how closely the character tracks the reference motion, r_{task} is a task-specific reward, and $w_{\text{imit}} = 0.5$ and $w_{\text{task}} = 0.5$ are their weights. The compliant motion controller describes not

only the joint angles but also the movements of the root joint. The imitation reward r_{imit} is defined by the multiplication of two terms:

$$r_{\text{imit}} = r_q r_p \tag{5.12}$$

where r_q is the pose reward and r_p is the positional reward. The pose reward measures a discrepancy between the simulated character pose and the reference pose, and the positional reward encourages the character to move in accordance with the root displacement $\mathbf{u}_p(t)$. The positional goal can be achieved by biped locomotion, which requires motion planning such as a motion graph [109], an RNN-based motion generator [108], motion matching [112], or a deep-learning-based high-level controller [53]. These techniques explicitly calculate the difference between the simulated pose and the reference pose frame-by-frame and use it to construct the imitation reward. Complex planning of motion sequences is required to achieve the positional objective, and thus the positional reward highly depends on the agility and the quality of the motion planner. Alternatively, we utilize a generative adversarial network for motion planning. Given an unstructured dataset of motions, we train an adversarial motion network to discriminate the simulated character pose from the reference motion distribution. Once trained, the network can evaluate how similar the character pose is to reference data, acting as the imitation reward function. The ability to learn the motion distribution enables the controller to actively plan goal-driven maneuvers along the distribution of the reference data without using any pre-generated trajectories.

To learn the adversarial motion network, we follow the trails of generative adversarial imitation learning, which stems from behavior cloning [113, 114, 115]. We first extract expert trajectories from the motion dataset $\mathcal{M}$. The trajectories are represented as pairs of the current and next state $(\mathbf{s}, \mathbf{s}') \sim \mathcal{M}$

in a per-frame basis. While learning the controller, the controller π generates controller trajectories $(\mathbf{s}, \mathbf{s}') \sim \pi$. The adversarial network $D_\psi(\mathbf{s}, \mathbf{s}') \in \mathbb{R}$ is optimized to discriminate the controller trajectories from the expert trajectories:

$$\min_{\psi} \left[\mathbb{E}_{(\mathbf{s},\mathbf{s}')\sim\mathcal{M}}(D_\psi(\mathbf{s}, \mathbf{s}') - 1)^2 + \mathbb{E}_{(\mathbf{s},\mathbf{s}')\sim\pi}(D_\psi(\mathbf{s}, \mathbf{s}') + 1)^2 \right] \tag{5.13}$$

where we use a least-square loss to prevent the gradient vanishing problem [115]. The pose reward function r_q is given by:

$$r_q(\mathbf{s}, \mathbf{s}') = 1 - \alpha(D(\mathbf{s}, \mathbf{s}') - 1)^2 \tag{5.14}$$

where α is the shape factor. The collaboration of Equation (5.13) and (5.14) encourages the controller to produce the current and next state pairs $(\mathbf{s}, \mathbf{s}')$ that are indistinguishable from the expert trajectory's distribution. When the expert dataset is small, the discriminator learns to discern expert trajectories from the controller's motion too quickly, such that the discriminator reward becomes unhelpful for the controller to imitate the dataset. In this case we use $\alpha = 0.2$ to give an offset reward whereas we use $\alpha = 0.25$ for large datasets. The positional reward r_p is designed to match the positional displacement:

$$r_p = w_p \exp\big(-\sigma_p \|\mathbf{p} - \mathbf{u}_p\|^2\big) + w_v \exp\big(-\sigma_v \|\mathbf{v} - \dot{\mathbf{u}}_p\|^2\big) \tag{5.15}$$

where $\mathbf{p}$ and $\mathbf{v}$ are the position and the linear velocity of the root body respectively. $\mathbf{u}_p$ and $\dot{\mathbf{u}}_p$ are the positional displacement of the root joint and its velocity respectively, which we previously obtained from the compliance-induced motion controller. In our experiments, the weights are $w_p = 0.7$, $w_v = 0.3$, $\sigma_p = 0.5$, and $\sigma = 1.0$.

5.3.2 State

Similar to previous work, we specify the state [2, 109, 108, 116, 117]. The dynamic state $\mathbf{s}_{\text{dyn}}$ is expressed in terms of positions, orientations, linear velocities,

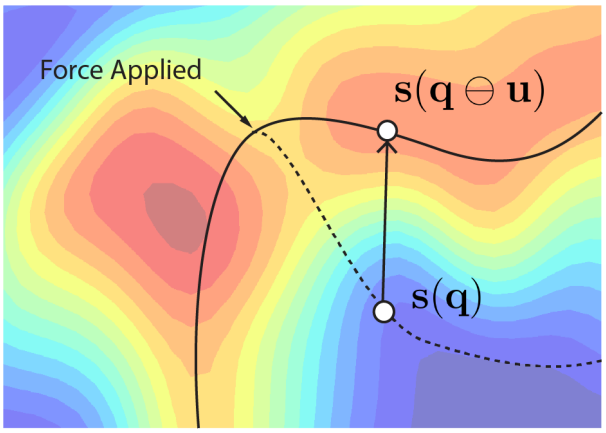


Figure 5.3: Expert motion distribution and the controller trajectories. The level-set represents how much the state is similar to the expert distribution. The expert trajectories are mainly located in the red area. The dotted line denotes controller trajectories while the solid line denotes trajectories of the shifted state. The shifted state is indistinguishable from the expert motion distribution.

and angular velocities of each body. The orientations are encoded using rotation matrices. The positions and the velocities are expressed in the character's local coordinate. The local coordinate is defined as the root body position projected onto the ground and the root body orientation aligned with the xz-plane. Overall, the state dimension is 306.

We slightly modify features of the state to accommodate for adversarial motion network training. We represent the state in a velocity-free manner to easily incorporate the motion displacement into the state features. We include positions of all bodies at the current character pose and the previous character pose. The positions are expressed in the character's current local coordinate.

As the imitation controller learns to displace its bodies according to the external forces, the controller trajectories deviate from the expert motion tra-

jectories by the amount of motion displacement $\mathbf{u}$ (see Figure 5.3). Learning the displaced expert motion $\hat{\mathbf{q}} = \mathbf{q}^m \oplus \mathbf{u}$ for all $m \in \mathcal{M}$ requires numerous sampling of the current and next state pairs $(\mathbf{s}, \mathbf{s}') \sim \hat{\mathbf{q}}$ and evaluating again the adversarial motion network for all $\mathbf{u}$. It is intractable to shift the expert motion distribution directly. We rather shift the state of the simulated character by the opposite direction of $\mathbf{u}$. We encode the state $\mathbf{s} = \mathbf{s}(\mathbf{q} \ominus \mathbf{u})$ where $\ominus$ computes the joint-wise quaternion differences $\mathbf{q} \ominus \mathbf{u} := (\log(\mathbf{u}_0^{-1}\mathbf{q}_0), \log(\mathbf{u}_1^{-1}\mathbf{q}_1), ..., \log(\mathbf{u}_n^{-1}\mathbf{q}_n))$, where n is the number of joints. By comparing the expert trajectories and the shifted state, we can construct an imitation reward that accomodates compliance.

The compliant motion controller evolves in time at a high frame rate (600Hz) while the imitation controller advances at a low frame rate (30Hz). Forces are unstable to integrate at a large timestep, and thus the motion controller requires a high frame rate in order to control the forces precisely. The imitation controller is responsible for long-term planning to achieve complex tasks and it works well with low frequency control. The motion controller works as a filter that converts the forces to the integrated form. The motion displacement $\mathbf{u}$ serves as an intermediary between two controllers, and the collaboration of two controllers enables the character to achieve both per-frame compliance and long-term goals.

5.4 Experimental Results

5.4.1 Environment and Controller Settings

Our simulation is written in C++. We use the open source library DART to simulate articulated rigid bodies [97]. Our character consists of 12 ball-and-socket joints and 2 welded joints. The character is 163cm tall and weighs 65kg. The shape of each body is modeled by a sphere or a box to compute the inertia

tensor and to detect collisions between the character and environments. We use PD gains $k_{\mathrm{p}} = 300$ to $500\ N \cdot m/rad$ and $k_{\mathrm{v}} = 2\sqrt{k_{\mathrm{p}}}$ for critical damping. Stable PD control is used to increase simulation stability [118]. Our simulation frequency is 600Hz, but when the external forces are large enough that it affects the simulation's stability, we simulate the dynamics at a higher rate of 900Hz.

The compliant motion controller starts with zero displacement and is integrated at the same rate as the physics simulation. Once the interaction forces are detected, we prepare the Jacobian matrix to compute the stiffness matrix. We sometimes give weights to the columns of the Jacobian matrix to generate task-dependent plausible motions. Specificallym we give more weights for the hip joints and the shoulder joints due to the anatomical property that the range of motions of those joints are wider than the other joints. When there are no interaction forces, the spring-damper system is activated on the motion displacement to recover the original motions.

Our imitation controller is based on a deep neural network. We use the open source framework PyTorch for learning the neural network [99]. We stack three fully connected layers for the controller $\pi_\theta(\mathbf{a}|\mathbf{s})$ as well as the value function $V^\pi(\mathbf{s})$. Each layer has 256 nodes and is initialized using Xavier initialization, and no dropout is used in our experiments. We use Proximal Policy Optimization(PPO) to learn the imitation controller. We use $\lambda = 0.95$ for the Generalized Advantage Estimate(GAE) and the discount factor $\gamma = 0.95$. During learning, a Gaussian noise for each action space is used for exploration, and we set the standard deviations to be 6°. Whenever 2048 state transition tuples $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ are collected, we update the parameters of the controller using a stochastic gradient descent method at learning rate 10^{-5} with a minibatch size of 128. The learning takes 6 to 24 hours with 20 to 80 million tuples for challenging tasks such as performing a backflip. We use a Ryzen 3950x CPU equipped with

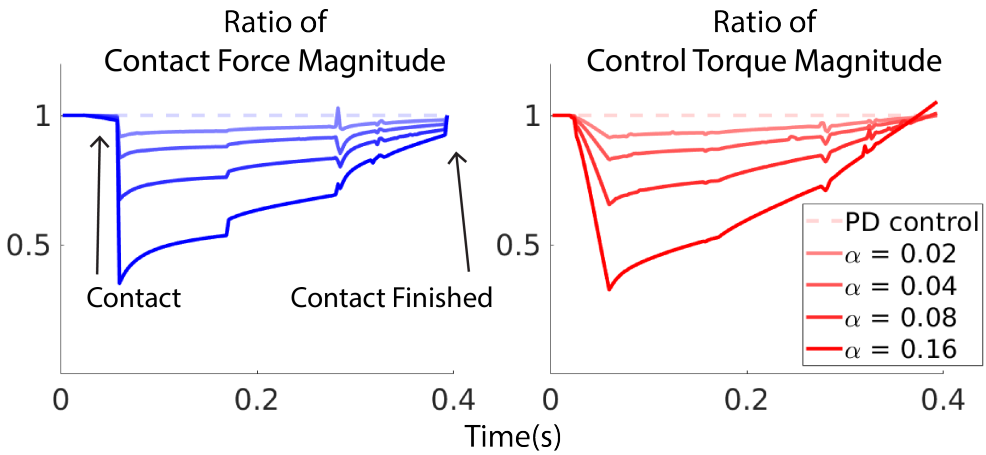


Figure 5.4: Levels of compliance. (Left) The ratio of contact force magnitude when the character collides with an obstacle. (Right) The ratio of control torque magnitude. Both are normalized by the values of standard PD controlled character without compliant motion control.

MPI-based parallelization to simulate the physics environment and collect the state transition tuples. NVIDIA 2070Ti GPU is used to accelerate the training of neural networks.

We also train a discriminator to evaluate the reward function in DRL. The discriminator and the imitation controller are updated jointly. The discriminator consists of three fully connected layers with 256 nodes. To learn the discriminator, we sample the controller trajectories $(\mathbf{s}, \mathbf{s}') \sim \pi$ from the simulator as well as the expert trajectories from the reference motions $(\mathbf{s}, \mathbf{s}') \sim \mathcal{M}$. We sample an equal amount of tuples $(\mathbf{s}, \mathbf{s}')$ from the simulation and the dataset in order for the discriminator to learn two distributions in a balanced manner. We use a minibatch size of 16 with learning rate 10^{-4}. In addition to the loss function (5.13), we add a gradient penalty loss term and a regularization loss term for the parameters of the last layer to enhance learning stability [119].

Table 5.1: Task-specific motion lengths and the levels of compliance.

Task	**Motion**	**Length**	**Level of Compliance**
Push recovery	Stand, Walk	54.6s	0.16
Interactive Control	Stand	0.03	[0.04, 0.16]
Hand-in-hand Run	Stand, Walk, Run	138.0s	[0.08, 0.16]
Chicken Fight	Chicken hopping	17.6s	0.16
Trampoline	Stand, Backflip	4.0s	[0.04, 0.08]
Balancing a ball	Stand, Walk	54.6s	0.16
Opening Doors	Opening a door	5.0s	0.16

5.4.2 Level of Compliance

We choose the level of compliance $\alpha = [0.04, 0.16]$ depending on tasks and interaction forces (see Table 5.1). To determine a good starting point for this value, we conduct a simple experiment for our motion control with various levels of compliance. In this experiment, the character is pushed by obstacles and the character learns to maintain its balance. The obstacles weigh 1kg and are thrown in various directions with initial velocities $[2.0, 3.0] m/s$. During contacts, both the magnitude of contact forces and magnitude of torques required for PD control decrease as the level of compliance increases (see Figure 5.4). We observe that the character's joint angles produce an implausible range of motion when $\alpha > 0.2$. In our later experiments, we mostly use $\alpha = 0.16$. When the interaction forces become larger, we decrease the level of compliance until it does not exceed the range of motion.

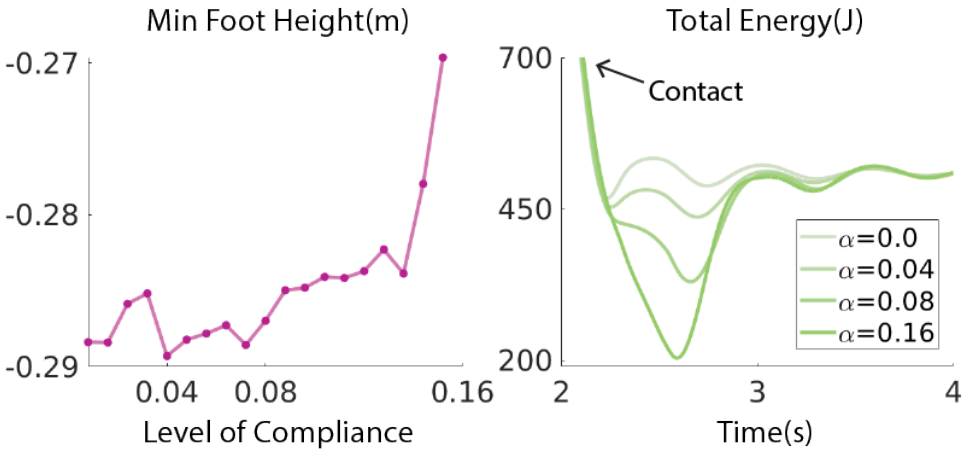


Figure 5.5: The *Trampoline* example. (Left) Minimum foot height during contact. Higher value indicates lower fluctuation of the trampoline. (Right) Total energy.

5.4.3 Learning Motor Skills

We learn assorted motor skills with our compliant controller. We use reference motions available on the web such as CMU motion dataset and *Mixamo*. The motion clips are retargeted to our character using *Autodesk MotionBuilder*™.

Hand-in-hand running is a task where two characters run together hand in hand. The hands are attached using ball-and-socket joints and thus forces are continuously applied along the hands. We prepare a pulling character which moves kinematically. We do not simulate the pulling character, but it serves as a kinematic constraint for the simulated character. While adhering to the kinematic constraint, the simulated character successfully learns to take additional steps to prevent losing its balance. In addition, the compliant control reduces the interaction forces. For a side-by-side comparison, we show a character that learns motor skills without using our compliant motion control. Since small changes in movement create drastically changing constraint forces, the charac-

ter moves dominantly by the constraint forces and produces stiff movements.

Chicken Fight is a sport where two or more players hop on one leg and bump the opponent players until they fall over. We use 17.6 seconds of hopping motion which moves in all directions. We learn two players with different mass distributions. One weighs 85kg and the other weighs 36kg. We modulate PD gains according to the different masses. According to Newton's third law of motion, they receive the same magnitude of forces when bumping. Since the mass distribution affects the forward dynamics and stiffness, the light character is pushed out farther away than the heavy character to compensate for the changes in momentum.

Trampoline is a device consisting of stretchable fabric attached to a steel frame. Humans leverage spring forces of the fabric to bounce on the trampoline. We learn motor skills on the trampoline ranging from balancing to backflip. To simplify the simulation, the trampoline is modeled as a rigid plate with linear actuators at each corner. We vertically bounce the character every 3 seconds. Interestingly, our compliant controller dissipates the character's total energy, meaning that the character absorbs the impact during contact and reduces fluctuation of the trampoline (see Figure 5.5).

5.4.4 Manipulation

The character also learns to manipulate objects. During manipulation, it is crucial to repeat a dynamic process of sensing interaction forces and actuating bodies to adjust to the situations. Since manipulation tasks require much more active control from the character, we drive the character by designing additional task-specific reward functions.

Opening doors are one of the common physical interactions in daily life. Humans open a door by grabbing and pushing a knob. Since the door consists

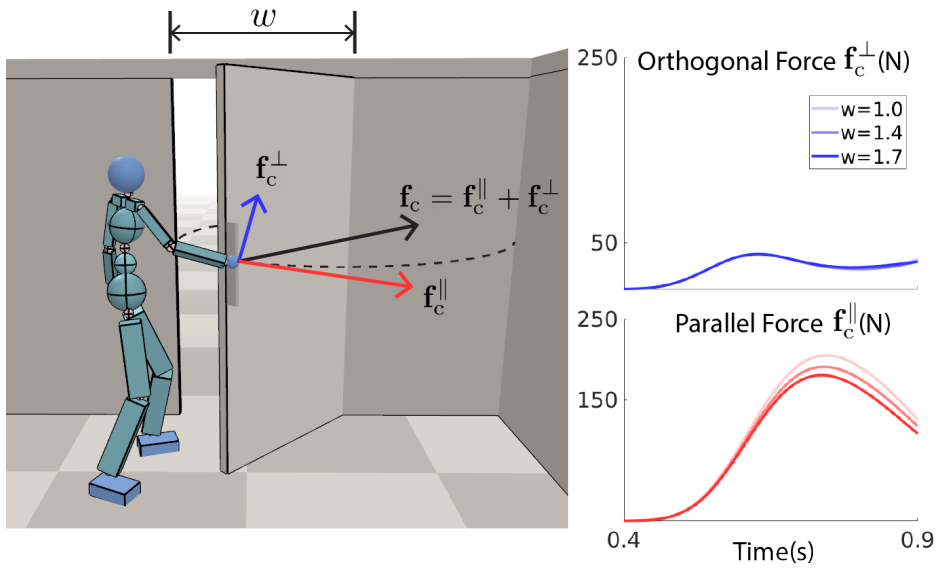


Figure 5.6: *Opening doors* example. (Left) The constraint force can be decomposed to orthogonal and parallel components relative to the knob, shown in blue and red arrows. (Up Right) The orthogonal force with the varying door size. (Bottom Right) The parallel force.

of a hinge, the manipulability of the knob forms a circle whose center is on the hinge. To open the door, the end-effector of the character should be on the circle. While forces acting parallel to the knob's normal direction works to open the door, forces acting perpendicular to the normal direction hinder the character from acting compliantly (see Figure 5.6). In our experiment, the character learns to open doors with varying door width w. When the hand is close to the knob, the hand is temporarily glued to the door using a zero-dof joint constraint. We take the orthogonal term of the constraint force as an input for the compliant control. The state for the task encodes the angle of the door and the relative position of knob expressed in the end-effector coordinate $\mathbf{s}_{\text{task}} = (\phi, \mathbf{p}_{\text{knob}})$. The task reward is designed as $r_{\text{task}} = \exp(-1.5 \max(0, \hat{\phi} - \phi))$ where $\hat{\phi} = 0.7$ is target door angle. After training, the character is able to deal with arbitrary size of the door ranging $w \in [1.0, 1.7]$ even though it only learns for the fixed size door $w = 1.0$ during learning. The compliant control successfully maintains the orthogonal forces consistently low independent of the door size while the character modulates the parallel forces according to the door size.

Balancing a ball is another example where the character learns to hold a ball on a rod on the character's head. Unlike other examples where the character passively receives forces, balancing the ball requires active procedures that produce forces to manipulate the ball as it wants. To do so, we design additional control variables in DRL. The control variables are 6-DOF which are the ghost forces at the end-effector and the root joint of the character. We encode the state for the task $\mathbf{s}_{\text{task}} = (\mathbf{p}_{\text{ball}}, \mathbf{v}_{\text{ball}}, \mathbf{p}_{\text{rod}}, \mathbf{v}_{\text{rod}})$ where $\mathbf{p}_{\text{ball}}, \mathbf{v}_{\text{ball}}$ are the position and the linear velocity of the ball and $\mathbf{p}_{\text{rod}}, \mathbf{v}_{\text{rod}}$ are the position and the linear velocity of the rod. These are expressed in the character's local coordinate. The task reward $r_{\text{task}} = \exp(-2.0||\text{proj}_{xz}(\mathbf{p}_{\text{ball}} - \mathbf{p}_{\text{head}})||^2)$ is designed to minimize the discrepancy between the position of the ball and the head pro-

jected onto the ground. As a result, the character actively deforms the pose and steps to move its position, mimicking the control of an inverted pendulum on a cart.

5.5 Discussion

Deep reinforcement learning with imitation rewards has gained great attention in physics-based character animation. Our framework enables reinforcement learning to address both long-term plans for imitation and short-term plans for compliance. The compliant controller reduces both the interaction forces and the control torques necessary for the balance and tasks. Moreover, our framework has the ability to learn manipulation tasks, which require active alternations of given reference motion sequences.

Our framework bridges the gap between position control and force control. Position control such as PD control is thought to be an indispensable module for the control of a high-dimensional floating-base character. Major drawbacks of using position control arise when the character interacts with its surroundings. The character applies a large amount of force to the environment, which potentially causes simulation instabilities or undesirable movements. Meanwhile, studies related to force control in Robotics have aimed on fixed-base low dimensional manipulators such as gripper robots. Our framework generalizes the concept of force control in terms of the character's compliance to incorporate force control into DRL. Combining extrinsic imitation of the human movements with intrinsic measurement of compliance allows the character to seamlessly learn both position control and force control.

Our framework also has limitations. We defined compliance assuming that it only relies on the character pose and its articulations, which are geometric

properties of the character. Indeed, human compliance is more complex than just geometry and articulation. For example, humans employ three strategies for maintaining their balance: ankle strategy, ankle-hip strategy, and stepping strategy. These strategies use extra information about the dynamics of the system, such as the center of pressure and base of support, which are not incorporated into our compliant control. Another limitation is that the method requires manual tuning of the level of compliance and the weights of the Jacobian matrix columns. It is feasible to tune the parameters by hand since the dimension of the parameters is relatively low. A straightforward expansion of our framework would consider anisotropic and joint-wise properties, which dramatically increases the dimension of hyperparameters. It might be possible to find these parameters from the motion dataset to reproduce realistic human movements.

Chapter 6

Conclusion

Understanding, analyzing and reproducing human movements have been a long standing goal in Computer Graphics, Robotics and Biomechanics. We simulate and control musculoskeletal system to manifest the functionalities of the motor system, as well as we demonstrate the sensorimotor control driven by the dynamics process of sensing forces and actuating bodies according to the tactile stimulus.

This thesis steps towards the ultimate goal of a true digital replica of the human and its movements. We incorporate volumetric muscles into our system to imitate a real mechanics of muscle contraction dynamics. The volumetric muscles represent much wider space of human body conditions than the muscles with the line-segment primitives. We exhibit such representation power with various muscle deficits such as muscle weakness, paralysis, and contracture. We also embed muscle-induced joint limit into our framework to provide reliable joint range of motions, and it produces variants of biped locomotions according to the muscle conditions even including pathological ones as well.

Finally the adoption of the concept of force control produces physically reliable motions when tactile interactions, which minimizes interaction impacts and control torques. To reproduce realistic human movements, not only we require precise modeling of humans, but also we need to suitable movement mechanisms that incur physiologically reliable motions.

Apparently, the frameworks reproduce human motions in detail as we provide more DOFs for the human models. For examples, the two heads of the biceps brachii are mechanically coupled in our system by the virtue of both being embedded in the same deformable tetrahedral mesh. As a consequence, contraction of one head would incur a mechanical forces on the endpoints corresponding to the other head, as volumetric muscles have shown its functionality for simulating real biological system. Our multi-segment foot model has additional 24 DOFs and 60 muscles to add subtle, yet important details for gait analysis. The extra DOFs allows the model to mimic softtissue structures of the foot, which contributes to absorb ground reaction forces when walking, running and kicking. In active manipulation as in *Balancing a ball* in Chapter 5, we provide additional DOFs in the control spaces, allowing the control policy to actively deform the body to its preference. The resulting motions imitate the balance control of an inverted pendulum on a cart.

We can think of many applications that exploit our framework:

- *Surgery Planning*: Our framework has potential to plan surgeries with the virtue of predicting pre-operative and post-operative gaits. 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.

An orthopedic surgery is such a precise operation to a specific body part, which incurs our framework limitations. To carefully imitate the surgery, we require accurate modeling of muscles, ligaments, tendons, nerves, and cartilages, and their interplays such as muscle-bone contact and muscle-muscle contact as well. Our volumetric muscle simulation has shown its potential to step towards highly-detailed human models, yet it is still technically challenging to incorporate high-dimensional non-linear muscles into full-body musculoskeletal simulation. Simulation of cutting, breaking and lengthening of muscles and bones is yet another technical issue that needs to be addressed.

- *Exoskeleton Testbed*: Our framework can be a testbed for designing exoskeletal robots, and optimizing their actuation parameters. In the quest to leverage intelligent robots that involve in physical human-robot interactions, learning from physically simulated humans and environments enables robots to safely learn from failure without putting real people at risk. Our framework as a testbed would be effective for designing robot such as the way of routing wired exoskeletal robot, and optimizing robot actuation mechanisms that vary depending on the human body conditions and the following movement patterns.

  Exoskeletal robots physically support humans with additional force profiles, and this support forces can be thought to be external perturbative forces when learning human control policies, which degrades the function of the exoskeletons. Our compliant control algorithm can be a useful solution and incorporating it into our scalable musculoskeletal simulation

framework can provide meaningful observations for designing, modeling and optimizing exoskeletal robots.

- *Performance improvement*: There is a need in Sports for an assistant tool that improves athlete performance. Our framework provides in depth motor skills equipped with the accurate muscle contraction dynamics. We envision that our framework provides automatic posture correction to enhance physical ability such as jumping, lifting a bar, and throwing a ball, by analyzing the athlete's movements. We even think of planning muscle strength training, which requires not only learning relationship between movements and the performance, but also learning how the physical capabilities of each individual affect the performance.

  Even though our framework demonstrates its potential to analyze human movements, there remains several future work in terms of computational costs. Our framework generates motions in specific conditions, and it needs to learn the control policy again if the body conditions change. It takes at least a day for each condition, and thus its inverse problem which finds body conditions given motions takes intractable computational cost. Learning integrated control policy of human movements eligible for continuous spectrum of anotamical conditions would be an important cornerstone for understanding the relationship between motions and body conditions.

Main limitation of this thesis is verification and validation. Even though we did quantitative comparison between EMG data and simulation results for biped locomotion, we observe that its generalization for motions other than biped locomotion to validate all the functionality of muscles as well as skeletal structures is not obvious propositions. This is because not only our model pa-

rameters were not finely tuned to synchronize to real world human, but also it is sometime intractable to acquire appropriate data for comparison. We envision a validation loop that our framework serves for surgery planning for patients, a testbed for exoskeletal robots, and performance improvement for athletes in terms of cost, efficiency, and data, and reversely, these applications verify and validate our framework by providing relavant data. These processes would be an important cornerstone for achieving the high-quality musculoskeletal simulation.

Bibliography

[1] Y. Jiang, T. Van Wouwe, F. De Groote, and C. K. Liu, "Synthesis of biologically realistic human motion using joint torque actuation," *ACM Transactions On Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[2] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[3] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, "Projective dynamics: fusing constraint projections for fast simulation," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[4] F. E. Zajac, "Muscle and tendon properties models scaling and application to biomechanics and motor," *Critical reviews in biomedical engineering*, vol. 17, no. 4, pp. 359–411, 1989.

[5] D. G. Thelen *et al.*, "Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults," *Transactions-American Society Of Mechanical Engineers Journal Of Biomechanical Engineering*, vol. 125, no. 1, pp. 70–77, 2003.

[6] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, T. John, E. Guendelman, and D. G. Thelen, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

[7] M. Damsgaard, J. Rasmussen, S. T. Christensen, E. Surma, and M. D. Zee, "Analysis of musculoskeletal systems in the anybody modeling system," *Simulation Modelling Practice and Theory*, vol. 14, no. 8, pp. 1100 – 1111, 2006.

[8] J. E. Lloyd, I. Stavness, and S. Fels, "Artisynth: a fast interactive biomechanical modeling toolkit combining multibody and finite element simulation," 2012.

[9] A. L. Cruz Ruiz, C. Pontonnier, N. Pronost, and G. Dumont, "Muscle-based control for character animation," *Computer Graphics Forum*, vol. 36, no. 6, pp. 122–147, 2017.

[10] J. Teran, S. S. Blemker, V. Ng-Thow-Hing, and R. Fedkiw, "Finite volume methods for the simulation of skeletal muscle," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 68–74, 2003.

[11] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw, "Creating and simulating skeletal muscle from the visible human data set," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 317–328, 2005.

[12] J. Stelletta, R. Dumas, and Y. Lafon, "Modeling of the thigh: a 3d deformable approach considering muscle interactions," in *Biomechanics of Living Organs*, pp. 497–521, 2017.

[13] S. Lee, E. Sifakis, and D. Terzopoulos, "Comprehensive biomechanical modeling and simulation of the upper body," *ACM Transactions on Graphics*, vol. 28, no. 4, 2009.

[14] W. Si, S.-H. Lee, E. Sifakis, and D. Terzopoulos, "Realistic biomechanical simulation and control of human swimming," *ACM Transactions on Graphics*, vol. 34, no. 1, 2014.

[15] S. H. Lee and D. Terzopoulos, "Heads up!: Biomechanical modeling and neuromuscular control of the neck," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1188–1198, 2006.

[16] T. Komura, Y. Shinagawa, and T. L. Kunii, "Creating and retargetting motion by the musculoskeletal human body model," *The Visual Computer*, vol. 16, no. 5, pp. 254–270, 2000.

[17] S. Sueda, A. Kaufman, and D. K. Pai, "Musculotendon simulation for hand animation," *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[18] P. Sachdeva, S. Sueda, S. Bradley, M. Fain, and D. K. Pai, "Biomechanical simulation and control of hands and tendinous systems," *ACM Transactions on Graphics*, vol. 34, no. 4, 2015.

[19] Y. Fan, J. Litven, and D. K. Pai, "Active volumetric musculoskeletal systems," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[20] M. Nakada, T. Zhou, H. Chen, T. Weiss, and D. Terzopoulos, "Deep learning of biomimetic sensorimotor control for biomechanical human animation," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[21] E. Sifakis, I. Neverov, and R. Fedkiw, "Automatic determination of facial muscle activations from sparse motion capture marker data," *ACM Transactions on Graphics*, vol. 24, no. 3, 2005.

[22] A.-E. Ichim, P. Kadleček, L. Kavan, and M. Pauly, "Phace: physics-based face modeling and animation," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[23] Y. Kozlov, D. Bradley, M. Bächer, B. Thomaszewski, T. Beeler, and M. Gross, "Enriching facial blendshape rigs with physical simulation," *Computer Graphics Forum*, vol. 36, no. 2, pp. 75–84, 2017.

[24] L. Zhu, X. Hu, and L. Kavan, "Adaptable anatomical models for realistic bone motion reconstruction," *Computer Graphics Forum*, vol. 34, no. 2, pp. 459–471, 2015.

[25] S. Saito, Z.-Y. Zhou, and L. Kavan, "Computational bodybuilding: Anatomically-based modeling of human bodies," *ACM Transactions on Graphics*, vol. 34, no. 4, 2015.

[26] P. Kadleček, A.-E. Ichim, T. Liu, J. Křivánek, and L. Kavan, "Reconstructing personalized anatomical models for physics-based body animation," *ACM Transactions on Graphics*, vol. 35, no. 6, 2016.

[27] I. Akhter and M. J. Black, "Pose-conditioned joint angle limits for 3d human pose reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1446–1455, 2015.

[28] Y. Jiang and C. K. Liu, "Data-driven approach to simulating realistic human joint constraints," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1098–1103, IEEE, 2018.

[29] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw, "Robust quasistatic finite elements and flesh simulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 181–190, 2005.

[30] T. Patterson, N. Mitchell, and E. Sifakis, "Simulation of complex nonlinear elastic bodies using lattice deformers," *ACM Transactions on Graphics*, vol. 31, no. 6, 2012.

[31] N. Mitchell, M. Aanjaneya, R. Setaluri, and E. Sifakis, "Non-manifold level sets: A multivalued implicit surface representation with applications to self-collision processing," *ACM Transactions on Graphics*, vol. 34, no. 6, 2015.

[32] N. Mitchell, E. Sifakis, *et al.*, "Gridiron: An interactive authoring and cognitive training foundation for reconstructive plastic surgery procedures," *ACM Transactions on Graphics*, vol. 34, no. 4, 2015.

[33] R. Malgat, B. Gilles, D. I. Levin, M. Nesme, and F. Faure, "Multifarious hierarchies of mechanical models for artist assigned levels-of-detail," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 27–36, 2015.

[34] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black, "Dyna: A model of dynamic human shape in motion," *ACM Transactions on Graphics*, vol. 34, no. 4, 2015.

[35] M. Kim, G. Pons-Moll, S. Pujades, S. Bang, J. Kim, M. J. Black, and S.-H. Lee, "Data-driven physics for human soft tissue animation," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 54:1–54:12, 2017.

[36] A. Murai, Q. Y. Hong, K. Yamane, and J. K. Hodgins, "Dynamic skin deformation simulation using musculoskeletal model and soft tissue dynamics," *Computational Visual Media*, vol. 3, no. 1, pp. 49–60, 2017.

[37] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović, "Interactive skeleton-driven dynamic deformations," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 586–593, 2002.

[38] L. Liu, K. Yin, B. Wang, and B. Guo, "Simulation and control of skeleton-driven soft body characters," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.

[39] H. Xu and J. Barbič, "Pose-space subspace dynamics," *ACM Transactions on Graphics*, vol. 35, no. 4, 2016.

[40] J. E. Lloyd, I. Stavness, and S. Fels, "Artisynth: a fast interactive biomechanical modeling toolkit combining multibody and finite element simulation," 2012.

[41] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "Sofa: A multi-model framework for interactive physical simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, vol. 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pp. 283–321, 2012.

[42] I. Stavness, A. G Hannam, J. E Lloyd, and S. Fels, "Predicting muscle patterns for hemimandibulectomy models," pp. 483–91, 2010.

[43] K. Yin, K. Loken, and M. van de Panne, "Simbicon: Simple biped locomotion control," *ACM Transaction on Graphics*, vol. 26, no. 3, 2007.

[44] K. W. Sok, M. Kim, and J. Lee, "Simulating biped behaviors from human motion data," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[45] Y. Lee, S. Kim, and J. Lee, "Data-driven biped control," *ACM Transactions on Graphics*, vol. 29, no. 4, p. 129, 2010.

[46] D. Han, J. Noh, X. Jin, J. S. Shin, and S. Y. Shin, "On-line real-time physics-based predictive motion control with balance recovery," in *Computer Graphics Forum*, vol. 33, pp. 245–254, 2014.

[47] J. M. Wang, S. R. Hamner, S. L. Delp, and V. Koltun, "Optimizing locomotion controllers using biologically-based actuators and objectives," *ACM Transaction on Graphics*, vol. 31, no. 4, 2012.

[48] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.

[49] Y. Lee, M. S. Park, T. Kwon, and J. Lee, "Locomotion control for many-muscle humanoids," *ACM Transactions on Graphics*, vol. 33, no. 6, 2014.

[50] Y. Lee, K. Lee, S.-S. Kwon, J. Jeong, C. O'Sullivan, M. S. Park, and J. Lee, "Push-recovery stability of biped locomotion," *ACM Transactions on Graphics*, vol. 34, no. 6, 2015.

[51] S. Coros, S. Martin, B. Thomaszewski, C. Schumacher, R. Sumner, and M. Gross, "Deformable objects alive!," *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[52] J. Tan, G. Turk, and C. K. Liu, "Soft body locomotion," *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[53] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[54] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[55] L. Liu and J. Hodgins, "Learning to schedule control fragments for physics-based characters using deep q-learning," *ACM Transactions on Graphics*, vol. 36, no. 3, 2017.

[56] L. Liu and J. Hodgins, "Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[57] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *ACM Transactions on Graphics*, vol. 37, no. 6, 2018.

[58] J. Won, J. Park, K. Kim, and J. Lee, "How to train your dragon: Example-guided control of flapping flight," *ACM Transactions on Graphics*, vol. 36, no. 6, 2017.

[59] J. Won, J. Park, and J. Lee, "Aerobatics control of flying creatures via self-regulated learning," in *SIGGRAPH Asia 2018 Technical Papers*, SIGGRAPH Asia '18, 2018.

[60] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk, "Learning to dress: Synthesizing human dressing motion via deep reinforcement learning," *ACM Transactions on Graphics*, vol. 37, no. 6, 2018.

[61] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[62] J. Merel, Y. Tassa, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, "Learning human behaviors from motion capture by adversarial imitation," *arXiv preprint arXiv:1707.02201*, 2017.

[63] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," in *Advances in Neural Information Processing Systems*, pp. 5320–5329, 2017.

[64] G. Berseth, C. Xie, P. Cernek, and M. Van de Panne, "Progressive reinforcement learning with distillation for multi-skilled motion control," *arXiv preprint arXiv:1802.04765*, 2018.

[65] J. Won, J. Park, and J. Lee, "Aerobatics control of flying creatures via self-regulated learning," *ACM Trans. Graph.*, vol. 37, Dec. 2018.

[66] Y.-S. Luo, J. H. Soeseno, T. P.-C. Chen, and W.-C. Chen, "Carl: Controllable agent with reinforcement learning for quadruped locomotion," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 38–1, 2020.

[67] S. Min, J. Won, S. Lee, J. Park, and J. Lee, "Softcon: Simulation and control of soft-bodied animals with biomimetic actuators," *ACM Trans. Graph.*, vol. 38, Nov. 2019.

[68] A. Levy, R. Platt, and K. Saenko, "Hierarchical reinforcement learning with hindsight," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[69] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549, 2017.

[70] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[71] Ł. Kidziński, S. P. Mohanty, C. Ong, Z. Huang, *et al.*, "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," *arXiv preprint arXiv:1804.00361*, 2018.

[72] X. B. Peng and M. van de Panne, "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, 2017.

[73] D. Driess, H. Zimmermann, S. Wolfen, D. Suissa, D. Haeufle, D. Hennes, M. Toussaint, and S. Schmitt, "Learning to control redundant musculoskeletal systems with neural networks and sqp: Exploiting muscle properties," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6461–6468, IEEE, 2018.

[74] N. Hogan, "Impedance Control: An Approach to Manipulation: Part I—Theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.

[75] R. J. Anderson and M. W. Spong, "Hybrid impedance control of robotic manipulators," *IEEE Journal on Robotics and Automation*, vol. 4, no. 5, pp. 549–556, 1988.

[76] G. Pratt and M. Williamson, "Series elastic actuators," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, pp. 399–406 vol.1, 1995.

[77] M. W. Spong, "Modeling and Control of Elastic Joint Robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, pp. 310–318, 12 1987.

[78] D. E. Whitney, "Historical perspective and state of the art in robot force control," *The International Journal of Robotics Research*, vol. 6, no. 1, pp. 3–14, 1987.

[79] C. Ott, R. Mukherjee, and Y. Nakamura, "Unified impedance and admittance control," in *2010 IEEE International Conference on Robotics and Automation*, pp. 554–561, 2010.

[80] C. Carignan, J. Tang, and S. Roderick, "Development of an exoskeleton haptic interface for virtual task training," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3697–3702, IEEE, 2009.

[81] T. Nef, M. Mihelj, G. Kiefer, C. Perndl, R. Muller, and R. Riener, "Armin-exoskeleton for arm therapy in stroke patients," in *2007 IEEE 10th international conference on rehabilitation robotics*, pp. 68–74, IEEE, 2007.

[82] K. Anam and A. A. Al-Jumaily, "Active exoskeleton control systems: State of the art," *Procedia Engineering*, vol. 41, pp. 988–994, 2012.

[83] Z. Li, Z. Huang, W. He, and C.-Y. Su, "Adaptive impedance control for an upper limb robotic exoskeleton using biological signals," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 2, pp. 1664–1674, 2016.

[84] E. Magrini, F. Flacco, and A. De Luca, "Control of generalized contact motion and force in physical human-robot interaction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2298–2304, 2015.

[85] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1010–1017, 2019.

[86] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8943–8950, 2019.

[87] V. B. Zordan and J. K. Hodgins, "Motion capture-driven simulations that hit and react," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 89–96, 2002.

[88] O. Arikan, D. A. Forsyth, and J. F. O'Brien, "Pushing people around," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 59–66, 2005.

[89] B. Stephens, "Humanoid push recovery," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 589–595, IEEE, 2007.

[90] Y. Lee, K. Lee, S.-S. Kwon, J. Jeong, C. O'Sullivan, M. S. Park, and J. Lee, "Push-recovery stability of biped locomotion," *ACM Transactions on Graphics*, vol. 34, no. 6, 2015.

[91] S. Lee, S. Lee, Y. Lee, and J. Lee, "Learning a family of motor skills from a single motion clip," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–13, 2021.

[92] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, pp. 181–ff, 1995.

[93] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[94] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[95] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[96] J. Lee, "Representing rotations and orientations in geometric computing," *IEEE Computer Graphics and Applications*, vol. 28, no. 2, pp. 75–83, 2008.

[97] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "DART: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

[98] E. M. Arnold, S. R. Ward, R. L. Lieber, and S. L. Delp, "A model of the lower limb for analysis of human movement," *Annals of biomedical engineering*, vol. 38, no. 2, pp. 269–279, 2010.

[99] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[100] F. E. Zajac, R. R. Neptune, and S. A. Kautz, "Biomechanics and muscle coordination of human walking: part ii: lessons from dynamical simulations and clinical implications," *Gait & posture*, vol. 17, no. 1, pp. 1–17, 2003.

[101] A. Wachter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[102] E. Sifakis and J. Barbic, "Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH 2012 Courses*, p. 20, 2012.

[103] T. Liu, S. Bouaziz, and L. Kavan, "Quasi-newton methods for real-time simulation of hyperelastic materials," *ACM Transactions on Graphics*, vol. 36, no. 3, 2017.

[104] H. Ralston, "Energetics of human walking," in *Neural control of locomotion*, pp. 77–98, 1976.

[105] A. Wachter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[106] B. Polster, *The Mathematics of Juggling*. 2003.

[107] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 28–36, 2003.

[108] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee, "Learning predict-and-simulate policies from unorganized human motion data," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[109] J. Won, D. Gopinath, and J. Hodgins, "A scalable approach to control diverse behaviors for physically simulated characters," *ACM Transactions on Graphics*, vol. 39, no. 4, 2020.

[110] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[111] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[112] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "Drecon: Data-driven responsive control of physics-based characters," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[113] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

[114] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[115] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics*, vol. 40, July 2021.

[116] S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.

[117] L.-K. Ma, Z. Yang, T. Xin, B. Guo, and K. Yin, "Learning and exploring motor skills with spacetime bounds," *Computer Graphics Forum*, vol. 40, no. 2, 2021.

[118] J. Tan, K. Liu, and G. Turk, "Stable proportional-derivative controllers," *IEEE Computer Graphics and Applications*, vol. 31, no. 4, pp. 34–44, 2011.

[119] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

요약

사람의 움직임에는 감각운동계의 많은 요소들이 관여한다. 이 학위논문에서는 사람의 움직임에 기여하는 내재적인 매커니즘을 재현하기 위해 사람의 근골격계와 촉각운동계를 시뮬레이션한다. 우리는 운동계를 변화함으로써 걷기, 뛰기뿐 아니라 병적 움직임을 포함한 다양한 동작들을 재현할 수 있다. 우리는 또한 촉각에서 오는 정보를 통해 다양한 움직임을 만들어 낼 수 있는데, 이 움직임으로 외력에 의한 반응뿐아니라 원하는대로 환경을 조작할 수있다.

이 학위논문에서는 3개의 주제를 다룬다. 사람의 근골격계는 200개가 넘는 뼈, 600개가 넘는 근육이 존재하는 큰 시스템이다. 첫번째로 우리는 근골격계 시뮬레이션의 확장성에 대해 다룬다. 두번째로 더욱 사실적인 근골격 시뮬레이션을 위해, 부피를 가지는 근육을 시뮬레이션 하는 방법론에 대해 다룬다. 마지막으로 우리는 인지, 사고, 행동하는 사람의 감각운동을 디자인하고 촉각에서 오는 정보를 통해 외부와의 상호작용에서 다양하고 자연스러운 움직임을 만들어 내고자 한다. 우리는 다양한 예제를 통해 우리의 프레임워크가 사람의 움직임을 이해하고 분석하고 재현할 수 있는 유용한 도구가 될 수 있음을 보여주고자 한다.

주요어: 컴퓨터 애니메이션, 물리 시뮬레이션, 물리기반 캐릭터 애니메이션, 심층 강화 학습, 근육 모델, 근골격계 모델, 힘 제어기, 임피던스 제어

학번: 2016-27526