TECHNISCHE UNIVERSITÄT MÜNCHEN Lehrstuhl für Echtzeitsysteme und Robotik

Physics-Based Modeling and Simulation of Musculoskeletal Robots

Steffen Wittmeier

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		UnivProf. DrIng. Alin Albu-Schäffer
Prüfer der Dissertation:	1.	UnivProf. DrIng. habil. Alois Knoll
	2.	UnivProf. Dr. Rolf Pfeifer (Universität Zürich, Schweiz)

Die Dissertation wurde am 18.12.2013 bei der Technische Universität München eingereicht und durch die Fakultät für Informatik am 21.04.2014 angenommen.

The advantage of the emotions is that they lead us astray, and the advantage of Science is that it is not emotional.

> — Oscar Wilde The Picture of Dorian Gray

ABSTRACT

In the past decade, a new class of tendon-driven robots has emerged which replicates living beings with an unprecedented level of detail. These so-called musculoskeletal robots are characterized by a set of unique features, such as muscle replicas where the mapping between the muscle forces and the joint torques depends on the robot posture or complex joints with many degrees-of-freedom. On the one hand, these features enable new applications for this class of robots, such as an artificial test-bed for the investigation of biologically inspired control strategies or as service and rehabilitation robots where the compliance of the muscular system of these robots increases the safety of human-robot interactions. On the other hand, however, these unique features also introduce new challenges both in hard- and software. One approach to tackle these challenges is via simulations where each problem can be investigated in isolation and in a well-defined environment. But unfortunately, simulating musculoskeletal robots is a challenge in itself, especially if traditional joint-space simulation approaches are employed as an explicit mapping between the muscle forces and the joint torques is required. Hence, the use of an alternative body-space approach, originally developed for computer games and animations and commonly known as physics-based simulations, is proposed.

In this dissertation, the applicability of the physics-based simulation approach to the class of musculoskeletal robots is investigated. Therefore, the modeling of skeletons and muscles within the physicsbased framework is introduced. Here, particular emphasis is put on extending the rigid-body dynamic algorithms of physics-based simulations with joint friction models as well as on the computation of the muscle kinematics, i.e. the muscle path. For the muscle kinematics, models of increasing complexity are considered. These models range from simple straight-line connections to wrapping surfaces that employ geometric primitives, namely spheres and cylinders, or meshes to simulate the wrapping of muscles on the bone surface. Moreover, a calibration routine is presented that employs an Evolution Strategy (ES) to minimize the simulation-reality gap of the parametric simulation models for both static robot postures and dynamic trajectories. Finally, the developed modeling and calibration techniques are evaluated against a physical robot. For this purpose, the musculoskeletal robot Anthrob is: (i) developed, (ii) modeled and (iii) calibrated. To further assess the accuracy of each of the presented muscle kinematics models, three Anthrob models are derived. While model I is based solely on straight-line muscles, model II and model III use cylindrical

and mesh wrapping surfaces to model the muscle wrapping on the bones of the robot, respectively.

Prior to calibration, model III clearly out-performs model I and II for the tested static robot postures and trajectories. After calibration, these errors are further reduced leading to final errors in the range of the robot repeatability for models II and III. Based on these results, it can be concluded that the physics-based simulation approach is a suitable tool for the simulation of musculoskeletal robots and that it has the potential to promote the development of this particular class of robots in the future.

In den vergangenen zehn Jahren hat sich eine neue Klasse von sehnengetriebenen Robotern herauskristallisiert, welche Lebewesen mit einem noch nie dagewesenen Detailgrad nachbilden. Diese sogenannten muskuloskeletalen Roboter zeichnen sich durch eine Reihe von einzigartigen Merkmalen aus. Diese sind z. B. Muskelimitate, bei denen die aus den Muskelkräften resultierenden Gelenkdrehmomente von der Pose des Roboters abhängen, oder komplexe Gelenke mit vielen Freiheitsgraden. Einerseits ermöglichen diese Merkmale neue Anwendungen für diese Art von Robotern wie z.B. als Testplattform für biologisch-inspirierte Regelungsalgorithmen oder als Roboter im Bereich der Service- und Rehabilitationsrobotik, wo die Nachgiebigkeit des Muskelapparats die Sicherheit von Mensch-Maschine-Interaktionen erhöht. Andererseits stellen die einzigartigen Eigenschaften auch neue Herausforderungen dar-sowohl im Bereich der Hard- als auch der Software. Ein Ansatz zur Lösung dieser Herausforderungen ist dabei der Einsatz von Simulationen, da hier jede Eigenschaft isoliert und unter definierten Rahmenbedingungen untersucht werden kann. Leider muss jedoch die Simulation von muskuloskeletalen Robotern selbst bereits als eine Herausforderung betrachtet werden, speziell wenn klassische Drehmoment-basierte Simulationsmethoden zum Einsatz kommen, da hier eine explizite Umrechnung der Muskelkräfte auf Gelenkdrehmomente notwendig ist. Aus diesem Grund wird ein alternatives Simulationsverfahren, welches unter dem Namen physics-based simulation bekannt ist und welches ursprünglich für Computerspiele und Animationen entwickelt wurde, als mögliche Lösung des Simulationsproblems vorgeschlagen.

In dieser Dissertation wurde die Eignung von *physics-based simulations* für die Simulation von muskuloskeletalen Robotern untersucht. Hierzu wurden Skelett sowie Muskelmodelle für *physics-based* Simulationsumgebungen entwickelt. Besonderes Augenmerk wurde dabei auf die Erweiterung der Festkörperdynamik-Algorithmen von Physik-Engines um Gelenkreibungsmodelle sowie auf die Berechnung der Muskelkinematik, d. h. des Muskelpfades, gelegt. Für die Berechnung der Muskelkinematik werden dabei verschiedene, in ihrer Komplexität variierende Modelle vorgestellt. Diese Modelle reichen von simplen Direktverbindungen, bei denen der Muskel als eine Gerade zwischen den zwei Ansatzpunkten modelliert wird, bis hin zu Oberflächenmodellen basierend auf geometrischen Primitiva oder Polygonnetzen, welche die Simulation des Muskelpfades auf Knochenoberflächen ermöglichen. Darüber hinaus wird eine Kalibrierungsroutine, basierend auf einer Evolutionsstrategie, vorgestellt, welche eingesetzt werden kann, um den Simulationsfehler der parametrischen Simulationsmodelle zu minimieren. Schlussendlich wurden die entwickelten Modellierungs- und Kalibrierungstechniken anhand eines Beispielroboters getestet. Hierzu wurde der muskuloskeletale Roboter Anthrob entwickelt, modelliert und kalibriert. Um dabei die Genauigkeit der entwickelten Muskelkinematikmodelle bewerten zu können, wurden drei Anthrob-Modelle abgeleitet. Während in dem ersten Modell die Muskeln ausschließlich durch Direktverbindungen modelliert wurden, kommen beim zweiten und dritten Modell Oberflächenmodelle basierend auf Zylindern bzw. Polygonnetzen zum Einsatz, um den Muskelpfad auf dem Knochen zu approximieren.

Betrachtet man zunächst die unkalibrierten Modelle so konnten mit dem dritten Modell, basierend auf Polygonnetzen, die besten Ergebnisse erzielt werden. Diese Abweichungen konnten durch die Kalibrierung weiter reduziert werden. So wurden für das zweite und dritte Modell Abweichungen im Bereich der Wiederholgenauigkeit des Roboters gemessen. Auf Basis der erzielten Ergebnisse kann gefolgert werden, dass der *physics-based* Simulationsansatz für die Simulation von muskuloskeletalen Roboter geeignet ist und daher möglicherweise in Zukunft einen wichtigen Beitrag zur Weiterentwicklung dieser Roboterklasse leisten kann.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

S. Wittmeier, M. Jäntsch, K. Dalamagkidis, M. Rickert, H. Marques, and A. Knoll. Caliper: A universal robot simulation framework for tendon-driven robots. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1063–1068, 2011

S. Wittmeier, M. Jäntsch, K. Dalamagkidis, and A. Knoll. Physics-based modeling of an anthropomimetic robot. In *Intelligent Robots and Systems (IROS)*, 2011 *IEEE/RSJ International Conference on*, pages 4148–4153, 2011

S. Wittmeier, A. Gaschler, M. Jäntsch, K. Dalamagkidis, and A. Knoll. Calibration of a physics-based model of an anthropomimetic robot using evolution strategies. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on,* pages 445–450, oct. 2012

S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntsch, K. Jovanovic, R. Knight, H. G. Marques, P. Milosavljevic, B. Mitra, B. Svetozarevic, V. Potkonjak, R. Pfeifer, A. Knoll, and O. Holland. Toward anthropomimetic robotics: Development, simulation, and control of a musculoskeletal torso. *Artificial Life*, 19(1):171–193, nov 2012

S. Wittmeier, M. Jäntsch, K. Dalamagkidis, A. Panos, F. Volkart, and A. Knoll. Anthrob – A Printed Anthropomimetic Robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 342–347, 2013

ACKNOWLEDGMENTS

First of all, I would like to thank Professor Alois Knoll for giving me the opportunity to work at his chair and, in particular, for being part of two extremely interesting EU projects on musculoskeletal robots, namely ECCEROBOT and Myorobotics. Furthermore, I would like to express my gratitude to Professor Rolf Pfeifer for being my external reviewer.

Many thanks also go to my colleagues Michael Jäntsch and Konstantinos Dalamagkidis as well as to Hugo Gravato Marques for their advice and constructive criticism during the past few years as well as for proofreading this thesis.

Last but not least, many thanks go to everybody at the chair of *Robotics and Embedded Systems* as well as to our scientific and industrial project partners. I enjoyed working with all of you.

CONTENTS

1	INTRODUCTION	1
	1.1 Motivation	1
	1.2 Contributions	4
	1.3 Units and Notation	6
	1.4 Readers' Guide	6
2	BACKGROUND	9
	2.1 Tendon-Driven and Musculoskeletal Robots	9
	2.2 Joint-Space Simulation of Forward Dynamics	16
	2.3 Simulation Tools for Musculoskeletal Robots	21
3	SKELETON MODELING	23
9	3.1 Bones	24
	3.2 Joints	25
)
4	MUSCLE MODELING	29
	4.1 Actuator Dynamics	30
	4.1.1 Brushed DC Motor Model	31
	4.2 Muscle Kinematics	34
	4.2.1 Straight-Line Muscle	35
	4.2.2 Muscle Via-Points	36
	4.2.3 Spherical Wrapping Surfaces	38
	4.2.4 Cylindrical Wrapping Surfaces	44
	4.2.5 Meshes as Wrapping Surfaces	47
	4.3 Series Elastic Element Modeling	52
	4.3.1 Linear Spring-Damper	52
	4.3.2 Models of Linear Viscoelasticity	53
	4.4 Muscle Friction	56
	4.5 Muscle Dynamics Algorithm	60
	4.6 Discussion	62
5	MODEL CALIBRATION	65
	5.1 Evolution Strategy	66
	5.2 Statics Calibration	69
	5.3 Dynamics Calibration	73
6	THE MUSCULOSKELETAL ROBOT ANTHROB	75
-	6.1 Skeleton	75
	6.2 Muscles	78
	6.3 Receptors	70 81
		51
7	ANTHROB MODELING	85

	7.1 7.2 7.3 7.4 7.5	Skeleton Modeling	85 86 87 96 97 98 99 100 101 101 108
8	ANT	HROB MODEL CALIBRATION	111
	8.1	Calibration	111
		8.1.1 Tier 1: Glenohumeral Joint Models	112
		8.1.2 Tier 2: Complete Robot Models	122
	8.2	Validation	130
	8.3	Discussion	133
9	CON	CLUSIONS	137
9	9.1	Summary	137 137
	9.2	Future Works	140
	-	L-TIME MOTION CAPTURE	-
Α	KEA A.1	Hardware Setup	143 143
	A.1 A.2	Calibration	145 145
	A.3	Image Processing Pipeline	145
		A.3.1 Image Acquisition	- 4 5 147
		A.3.2 Marker Segmentation	147
		A.3.3 Marker Triangulation	148
		A.3.4 Marker Target Pose Estimation	149
		A.3.5 Joint Angle Computation	151
	A.4	Performance Benchmarking	151
	A.5	Summary	153
в	LIN	EAR VISCOELASTICITY	155
	B.1	Stress and Strain	155
	B.2	Basic Models	156
С	ΑΝΤ	HROB MODELS	159
-	C.1	Skeleton Model	159
	C.2	Muscle Models	-55 161
	с.3	Simulation-Reality Gap	166
P		UDOD CALIDDATION	
D	ANT D.1	Simulation-Reality Gap	167 167
	D.1	omunuton many Oap	10/

	D.2 Calibration Parameters	167
Е	CALIPER: A ROBOTICS DEVELOPMENT ENVIRONMENT E.1 CALIPER Core	179 179 180
F	QT PIPELINE TOOLKIT (QPT)	183
LIS	ST OF FIGURES	187
LIS	ST OF TABLES	190
LIS	ST OF ALGORITHMS	191
LIS	ST OF ABBREVIATIONS	192
LIS	ST OF SYMBOLS	194
BII	BLIOGRAPHY	197

INTRODUCTION

1.1 MOTIVATION

Motor control and object manipulations are complex tasks that humans execute with remarkable ease, accuracy and repeatability. This immediately poses the question of how the brain is able to solve the underlying control problem of coordinating the vast number of muscles required for a particular movement. Today, this question is mainly raised by Neuroscientists who employ transection and/or lesioning experiments [138] as well as non-invasive recording techniques, such as Functional Magnetic Resonance Imaging (fMRI) [50] or Electromyography (EMG) [7], to elucidate the underlying neurophysiological processes of human motor control. Even though these studies and technologies have significantly contributed to our current knowledge of human motor control, many questions still remain unresolved. One possibility of shedding new light on these questions is to employ an interdisciplinary approach uniting robotics and neuroscience in accordance with the synthetic methodology principle "understanding by building" [162].

The study of artificial systems has already significantly contributed to our understanding of human motor control. For instance, industrial manipulators have been used to study the problems associated with the planning and execution of human limb trajectories [47]. But, robotic manipulators are very different from humans. First, the usage of high precision torque actuators located within joints is an oversimplification of the human muscular system where even single joint movements require the phased excitation and relaxation of multiple muscles. Second, robotic manipulators are traditionally composed of rigid links attached to single Degree of Freedom (DoF) revolute or prismatic joints [16], characteristics which facilitate control and improve manipulation performance. But the human body is neither stiff nor made exclusively of revolute joints. These missing properties pose severe limitations for the aspects of motor control that can be investigated by using industrial robots and on the conclusions that can be drawn from the experimental results. Therefore, more detailed replicas of the human motor apparatus are a requisite for the effective

Human motor control & neurorobotics

Limitations of traditional robots

2 INTRODUCTION

investigation of human motor control issues in artificial systems—in accordance with the concept of embodiment [115].

Some progress towards this goal has been made by the many hu-Anthropomorphic & musculoskeletal manoid robots that have been developed around the world, such robots as Asimo [130] or iCub [154], which can deliver impressive performance on the tasks that they are designed for (e.g. [14]). So far, however, these anthropomorphic robots have failed to generate a significant stream of new findings in the area of human motor control. Hence, a new class of robots has emerged in the past decade which has the potential to unite researchers from the fields of Neurosciences and Robotics and to finally reveal the substrates of human motor control. This class of robots is called musculoskeletal or anthropomimetic [46] robots. Musculoskeletal designs differ from anthropomorphic humanoid robots by not only imitating the morphological appearance of humans, but by capitalizing on the replication of the inner structures of the human body such as equivalents of compliant muscles, tendons, bones, and joints. The resulting robots are highly complex artifacts that belong to the super-class of tendondriven robots but exhibit a set of distinct features such as a pointto-point tendon routing (i.e. tendons that can only induce a unidirectional joint rotation) or redundant, antagonistic actuators with posture-dependent lever arms (i.e. lever arms that change with the joint positions). It is these human-inspired features which make musculoskeletal robots both unique and challenging.

Challenges of musculoskeletal robots

Even though musculoskeletal robots now exist for more than a decade, the scientific progress, particularly in the area of control, falls short of expectations. The main reason for this is that the unique features of these robots introduced new challenges, both in hard- and software. As an example, let us consider a single spherical joint as it is for instance used in musculoskeletal robots to mimic the glenohumeral joint of the human shoulder girdle. In contrast to the 1 DoF revolute joints typically used in industrial robots, spherical joints provide 3 rotational DoF. This difference, which might appear to be minor at first, has a huge impact. First of all, a spherical joint features no bearings and hence the joint ball has to be actively pressed into the socket by the muscular system. This in turn introduces an interdependence between the muscle and the joint friction forces as higher muscle tensions lead to higher contact forces between the ball and the socket of the joint. Secondly, most control approaches developed for musculoskeletal robots are feedback control laws that rely on joint position errors to compute the actuator inputs for the next control cycle. But, measuring the joint position of a spherical joint is an inherently difficult task and no intrinsic solutions are available on the market today. This impedes the testing and evaluation of developed controllers on real artifacts. Furthermore, expressing the joint position by a scalar,

as for revolute joints, is not possible for spherical joints due the three DoF. Hence, a singular-free representation, such as unit quaternions, has to be used which further complicates the utilization of standard control laws developed for industrial robots. Even though this example demonstrates some of the challenges arising from the kinematic differences between industrial and musculoskeletal robots, there are even more significant challenges due to the muscular system of musculoskeletal robots. First of all, in industrial robots the actuator torque can be directly mapped to a joint torque. This is not the case for musculoskeletal robots, where the muscular system applies forces to the links of the robot. Thus a mapping between muscle forces and joint torques has to be found to employ joint-space control approaches from classical robotics to the class of musculoskeletal robots. But this mapping is not trivial to derive, due to the posture-dependence of the lever arms and the redundant actuation (i.e. more than one muscle influences a single joint coordinate). Furthermore, similar to the human paragon, musculoskeletal robots often feature multi-articular muscles (i.e. muscles that span more than one joint) or muscles that wrap around the skeleton surface. These properties further complicate the modeling of musculoskeletal robots and the derivation of a mapping between the muscle-force and joint-torque space.

One approach to tackle the challenges of musculoskeletal robots is via simulations where each challenge can be investigated in isolation and in a well-defined environment. For instance, a spherical joint could be equipped with a virtual, frictionless bearing or a virtual sensor to first analyze other aspects of the musculoskeletal design. But unfortunately, simulating musculoskeletal robots is a challenge in itself due to the complexity of the musculo-skeletal interactions. Thus, even though various simulation tools are available for the simulation of mobile or industrial robots, no comparable tools exist for musculoskeletal robots. Here, normally tailored solutions are used that have been developed to simulate the dynamics of a particular robot (e.g. [33] or [124]). However, due to the close analogy of musculoskeletal robots with their biological counterpart, the problems associated with the simulation of this class of robots is very similar to the problems encountered in biomechanics. Here, simulation models are used to investigate the control and the mechanics of biological systems and two simulation toolboxes are particularly noteworthy: OpenSim [21] and Anybody. Both tools focus on models of the human body and provide a variety of muscle models to simulate the dynamics arising from the musculo-skeletal interactions. However, they operate in the joint-space which means that the dynamics of the skeletons are simulated by applying torques to the joints and thus an explicit mapping between the muscle forces and the joint torques is required. But this mapping can be tedious to derive for complex musculoskeletal robots as shown in detail in Chapter 2. Hence, an alternative body-space simSimulation of musculoskeletal robots

4 INTRODUCTION

ulation approach, known as physics-based simulations, is presented in this thesis.

Physics-based simulations originate from computer games and ani-Physics-based simulations mations where so-called physics-engines are used to create the illusion of a virtual world that complies with the laws of Newtonian dynamics. A variety of physics-engines has emerged in recent years, such as PhysX, ODE, Bullet Physics or Newton. All these engines offer a rich set of functionality such as rigid-body and constrained dynamics required for the simulation of the skeletons of musculoskeletal robots (for a detailed comparison see [8]). The major difference between physics-engines and simulation tools from biomechanics is that physics-engines operate in the body-space. This means that body positions and orientations are integrated over time rather than joint positions. This fundamental difference simplifies the simulation of musculoskeletal robots as an explicit mapping between muscles forces and joint torques is no longer required and muscle forces can be applied directly to rigid bodies at the tendon terminations.

1.2 CONTRIBUTIONS

Simulation tools for musculoskeletal robots Musculoskeletal robots are impressive artifacts that often feature more than 40 skeletal DoF and at least the same number of muscles. But, so far, it is this complexity which has also impeded the application of these robots to real-world scenarios. Therefore, the goal of this thesis is to develop modeling and simulation techniques to provide researchers with a tool for the investigation of musculoskeletal robots in a well-defined, virtual environment. These modeling and simulation techniques are implemented on physics-engines which are originally developed for computer games and which operate in the body-space (see above). It is shown that it is this body-space approach which simplifies both the simulation of musculoskeletal robots as well as the model definition by the researcher—when compared to traditional joint-space approaches.

Muscle kinematics modeling In order to simulate the muscles in the body-space domain, the computation of the muscle kinematics is extremely important as they provide two substantial quantities: (i) the muscle length and (ii) the muscle lines of action. Therefore, various muscle kinematics models were developed. These range from straight-line connections to wrapping surfaces based on geometric primitives and meshes. Even though similar muscle kinematics models can be found in the field of biomechanics [20, 22, 23, 34, 79, 159], there are still considerable differences to the models presented here. For example, in 2000 Garner and Pandy presented the "obstacle-set method for representing muscle paths in musculoskeletal models" which included models of spherical and cylindrical wrapping surfaces [34]. However, the presented models were not capable of simulating a change in the wrapping direction, neither were they able to handle multiple wrapping revolutions. Both problems were solved in this work by introducing a state machine and a revolution counter which contributed to the versatility and applicability of these models to a wider range of use-cases. Another difference between the presented models and the models used in biomechanics is that in biomechanics the goal is typically to approximate the muscle length. This muscle length is subsequently used to compute the resulting joint torque by partial differentiation of the muscle length with respect to the joint angles [23] (i.e. the muscle Jacobian, see Section 2.2). However, a similar approach is not feasible in physics-based simulation engines which operate in the body-space and hence require the lines of action of the muscle to be exactly computed. This becomes particularly important in the case of mesh wrapping surfaces. Here, previous studies suggested to approximate the geodesic path on the convex envelope by the shortest path on the mesh computed via the algorithm of Dijkstra to reduce the computation time [23]. Even though this approach results in a good approximation of the path length as shown in Section 4.2.5 and confirmed by other studies [148], large errors in the lines of action were observed. Hence, the obtained results show that exact geodesic algorithms should be favored over approximations if physics-engines are used.

Furthermore, muscle friction, which plays an important role in the muscle dynamics [58], was neglected in previous studies. Therefore, a muscle friction model based on the capstan equation was derived which is capable of modeling the effects of Coulomb friction on spherical obstacles within the muscle path. However, while being advantageous for the simulation of the muscle dynamics, the body-space approach also has disadvantages. For instance, the modeling of joint friction is straight-forward in the joint-space domain but difficult in the body-space. Therefore, a joint friction model is presented that exploits the error correction term of the constraint solver of physicsengines to model the effects of Coulomb friction.

The simulation-reality gap of musculoskeletal robot models was never investigated by previous studies. Thus, a detailed analysis of the accuracy of the developed models and of the physics-based simulation approach in general was conducted by measuring the model errors for static robot postures and dynamic trajectories. For this purpose the musculoskeletal robot Anthrob, which mimics the human upper limb, was developed and modeled by the presented techniques. To measure the joint position of the spherical glenohumeral joint of the robot, a high-speed motion-capture system was developed which deMuscle & joint friction modeling

Simulation-reality gap analysis & minimization

6 INTRODUCTION

livers joint positions with an update rate of $60\,\text{Hz}$ and a latency of $4.3\,\text{ms}$.

Model calibration Furthermore, a calibration procedure was developed to minimize the simulation-reality gap of musculoskeletal robot models. This calibration procedure uses a (μ, λ) -Evolution Strategy and a Gaussian-based, non-isotropic, self-adapting mutation operator to explore the search space. In general, the developed procedure can be applied to a large variety of input data and error measures. However, as an example, two objective functions were developed that can be used to calibrate the statics (i. e. the simulation-reality gap of equilibrium postures) and the dynamics (i. e. the simulation-reality gap of trajectories) of musculoskeletal robots that feature compliant, tendon-driven actuators.

Real-time simulations and online applications Finally, real-time simulation performance was neither an issue nor a goal in previous studies. Here, the presented models perform well allowing for the real-time simulation of small- and medium-scale robots. In the future, these real-time capabilities will facilitate new *online* applications of musculoskeletal robot simulations, for instance as an internal model for control.

1.3 UNITS AND NOTATION

The mathematical notation in this thesis is based on the guidelines of the International Organization for Standardization (ISO)¹. Hence, scalar variables and functions are set in italic, whereas vectors and matrices are set in bold lower-case and upper-case letters, respectively. Moreover, an additional *hat* on vectors is used to indicate unit vectors (e. g. \hat{f}) and, if required, the frame of reference of the vector is indicated by a preceding superscript (e. g. ${}^{A}f$). In the cases were units are relevant, SI units are used with one exception: angular positions and position errors are presented in degrees for the sake of comprehensibility.

1.4 READERS' GUIDE

The content of this thesis can be divided into two parts: (i) a theoretical part presenting physics-based modeling techniques for musculoskeletal robots and (ii) a result part in which the models from the first part are evaluated.

Part I: Theory Chapter 2, which introduces the theoretical part, discusses the major differences between musculoskeletal and the super-class of tendon-driven robots. Furthermore, existing musculoskeletal robots are presented and an overview of joint-space simulation approaches is given.

¹ see ISO 80000-2:2009 or ISO 31-11

In Chapter 3, the physics-based modeling of musculoskeletal robot skeletons is introduced by outlining the analogies of skeletons and articulated rigid bodies. Subsequently, in Chapter 4 the physics-based modeling of the muscular system is described. Finally, Chapter 5 concludes the theoretical part by presenting a calibration procedure that can be employed to reduce the simulation-reality gap of musculoskeletal robot simulation models.

The models and algorithms derived in part I are evaluated in part II of the thesis. Thus, the musculoskeletal robot Anthrob, presented in Chapter 6, was developed. The derivation of three physics-based Anthrob models as well as the analysis of their simulation-reality gap is presented in Chapter 7. Finally, the simulation-reality gap of all three models is minimized by means of model calibration in Chapter 8.

The thesis is concluded by a summary and future works prospects in Chapter 9.

Part II: Results

BACKGROUND

2.1 TENDON-DRIVEN AND MUSCULOSKELETAL ROBOTS

Today, most industrial robots are actuated via high-precision drives that are located within joints to reduce backlash and increase manipulation performance. Despite the popularity and the advantages of this approach for many applications, such as industrial assembly tasks, there are also disadvantages. For instance, the inclusion of the actuators in the kinematic structure of the robot increases the weight and therefore also the inertia of the manipulator-an aspect that is of particular importance in human-centered robot applications, such as medical or service robotics, where a possible impact can cause severe injuries or even death due to the high impact loads [177]. One possibility to minimize these effects is to employ additional sensors to anticipate impacts or to use more complex control strategies, such as joint torque control, to reduce the severity of inevitable collisions [44]. Another promising approach to tackle this problem is to remove the actuators from the moving parts of the robot and to use a transmission system to convey their forces/torques to the required points of application. This also bears the advantage of higher payload-to-weight ratios. One such transmission system that is widely used in robotics since the early 80's are tendons. The anatomic term tendon, that in robotics commonly refers to ropes, cables, belts, wires and other similar transmission mechanisms [76], has been introduced by the first robotic systems that used this class of transmission mechanisms to replicate human hands (e.g. [54]).

Nowadays, many tendon-driven robots exist that can be categorized based on a variety of properties. For instance, Jacobsen et al. classified tendon-driven robots with regard to the number of actuators per skeletal DoF into N, N + 1 and 2N configurations [55, 56]. Another classification has been introduced by Jyh-Jone Lee in 1991 who distinguished two types of tendon routing techniques that he called: (i) open-ended and (ii) endless tendon [76]. Even though both presented classifications are still valid today, there are additional properties that are not captured and that are of particular interest in the context of this work. Therefore, a more generic, three-tier categorization of

Why use tendons as transmission system?

A three-tier classification of tendon-driven robots

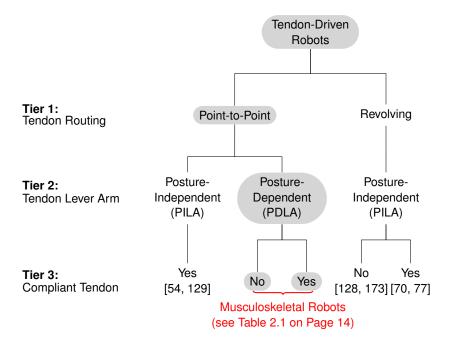


Figure 2.1: Three tier categorization of tendon-driven robots. Tendon-driven robots are categorized based on the tendon routing (Tier 1), the tendon lever arm (Tier 2) and the tendon compliance (Tier 3).

tendon-driven robots is introduced which extends the approach of Jyh-Jone Lee as shown in Figure 2.1. On the first tier, tendon-driven robots are distinguished based on two types of tendon routing in accordance with Lee. If the tendon forms a closed-loop and is able to induce a bidirectional joint rotation, it is called a *revolving* routing whereas tendons with two attachment points that only invoke a unidirectional joint rotation—similar to human muscles—are said to have a *point-to-point* routing. The differences between the two routing types are also shown in Figure 2.2.

The second tier classifies tendon-driven robots based on the lever arm properties of the tendons into (i) Posture-Independent Lever Arm (PILA) and (ii) Posture-Dependent Lever Arm (PDLA) tendons. In most early or industrial tendon-driven robots PILAs are used to simplify control as the joint torque can easily be back-computed from the measured tendon force taking the constant lever arm into account (see Figure 2.3a). However, biologically inspired robots, such as tendon-driven humanoids or quadrupeds, typically feature PDLAs, where the lever arm of an actuator is often a nonlinear function of the posture of the robot (see Figure 2.3b). The third and last tier classifies the tendons in stiff and compliant tendons. Stiff tendons again simplify the control whereas compliant tendons introduce a passive flexibility that increases the safety of these robots. Whether the tendon compliance is linear, nonlinear, inherent to the actuator technology as in pneumatic actuators or artificially introduced is not relevant at this point. The benefit of nonlinear compliance is that the

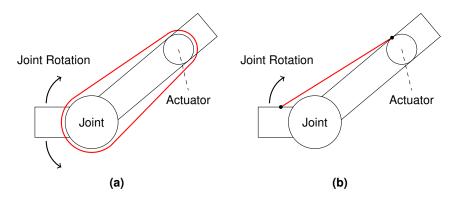


Figure 2.2: Revolving and point-to-point tendon routing. (a) Revolving tendon routing: the tendon can invoke a bidirectional joint rotation. (b) Point-to-Point tendon routing: the tendon can only invoke a unidirectional rotation of the joint-as in human muscles.

non-linearity can be exploited in antagonistically actuated joints to change the impedance of the joint by co-contracting the antagonistic actuators (e.g. to increase the stiffness when catching a ball or to decrease the stiffness when swinging the arm).

One of the first robotic systems that extensively made use of tendondriven actuation was the UTAH/M.I.T. Dexterous Hand, developed by Jacobsen et al. in 1986 as a research tool for the investigation of machine dexterity [54]. This hand, which was a joint development of the University of Utah and the Massachusetts Institute of Technology, featured 16 skeletal DoF that were actuated by 32 pneumatic, tendon-driven actuators in a 2N configuration. For the tendons, a point-to-point routing was used and additional pulleys were added to ensure that the lever arm of each tendon was posture-independent. Due to the pneumatic actuators, each tendon also featured a compliance that was exploited to adjust the joint stiffness through the cocontraction of the two antagonistic actuators. Another robotic hand, that uses a similar actuator setup, is the Shadow Dexterous Muscle Hand developed by the Shadow Robot Company [71, 129, 155]. Just as the UTAH/M.I.T. hand it uses pneumatic actuators to drive the 40 tendons and 24 skeletal DoF of the hand. A 2nd variant, the Shadow Dexterous Motor Hand, is also available. It uses 20 electromagnetic noncompliant actuators and a revolving tendon routing.

In general, revolving tendons have two main advantages: (i) the lever arm is posture-independent which facilitates control and (ii) less actuators are required which increases the payload-to-weight ratio of the robot. Two of the most important robots that use a revolving tendon are the BioRob-Arm, developed at the Technische Universität Darmstadt [70, 77], and ISELLA2 from the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA) [128]. The BioRob-Arm has four rotational DoF, each actuated by a single DC motor connected to a set of four series elastic tendons that are attached an-

Tendon-driven robotic hands

Robots with revolving tendons

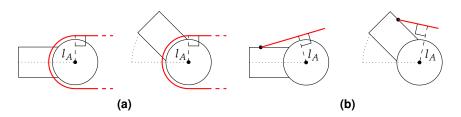


Figure 2.3: Posture independent and dependent lever arms. (a) Posture-Independent Lever Arm (PILA). The lever arm l_A is constant and does not depend on the joint position. (b) Posture-Dependent Lever Arm (PDLA). The lever arm l_A changes with the joint position.

tagonistically to the end of the actuated link [70]. The antagonistic routing allows for a more lightweight design of the links as it minimizes the bending stress. To further reduce the weight and inertia of the robot, the actuators are located at the base of the robot and are not mounted to the links. In summary, this smart design results in an extremely lightweight robot that has a high payload-to-weight ratio of ~ 0.5 and a high position accuracy of the end-effector of less than 1 mm. The Fraunhofer ISELLA2 is another example for a robot that uses a highly efficient actuation system based on a revolving tendon routing. Here, a specific rope actuator, the QuadHelix-Drive has been used to increase the payload-to-weight ratio [128]. The QuadHelix-Drive, which is an enhancement of the DOHELIX-Muscle also developed at the Fraunhofer IPA [145], uses four tendons that are routed via two turning wheels to induce joint rotations. The four tendons are stiff and are connected to a single DC motor shaft in such a way that two of the four tendons are coiled while the others are uncoiled when the shaft is rotated. To ensure a correct coiling of the tendons on the shaft, a worm gear is used to translate the motor unit and therefore the shaft during coiling and uncoiling. As the two tendon pairs are coiled/uncoiled at opposing sides of the shaft, the radial shaft-forces are balanced and no shaft bearing is required. Moreover, a shaft with a small diameter can be used which reduces the load torques of the actuator and facilitates the usage of small gear ratios while maintaining the capability of lifting heavy objects.

Musculoskeletal robots Another category of tendon-driven robots, which has become very popular in the last decade, are musculoskeletal robots. Musculoskeletal robots normally feature detailed replicas of the skeleton of living beings and employ actuators with a point-to-point tendon routing and posture-dependent lever arms to imitate muscles (see Figure 2.1). Even though these robots are typically built by roboticists, which seek to create artifacts that move and operate with the grace of biological systems, they are becoming of at least similar interest to neuroscientists and biomechanicists (e.g. for the study and testing of motor control theories). Particularly, a subclass of highly complex musculoskeletal robots has emerged that replicate the human skeletal and

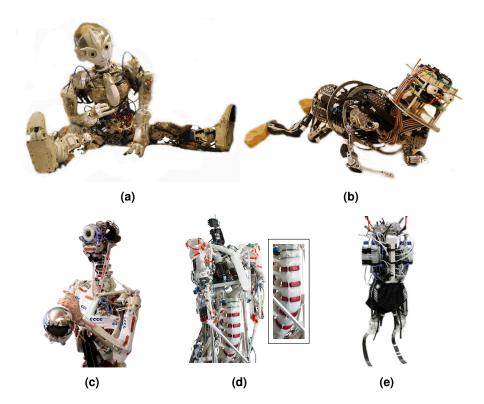


Figure 2.4: Examples of musculoskeletal robots. **(a)** Kotaro, a humanoid robot developed at the Jouhou System Kougaku Laboratory of the University of Tokyo. **(b)** Pneuborn-7II, a pneumatically actuated robot with a body corresponding to the size of a seven month old infant. **(c)** Ecce-2, a non-sensorized, but highly complex torso developed by the Eccerobot project. **(d)** Ecce-3, also developed by the Eccerobot project, which features an extremely detailed replica of the human spine. **(e)** Athlete Robot, a pneumatically actuated running robot developed at the University of Tokyo.

muscular system with an unprecedented level of detail and which are therefore sometimes referred to as *anthropomimetic* robots [46]. An overview of the most important musculoskeletal robots is given in Table 2.1.

The most complex musculoskeletal robots existing so far are built at the Jouhou System Kougaku Laboratory of the University of Tokyo (JSK) and by the project Eccerobot which was funded by the EU's 7th Framework Programme and involved five European universities, including the Technische Universität München. Both research groups focus on humanoid designs and use electromagnetic tendon-driven actuators as muscle imitations. The first robot built by JSK—Kenta was completed in 2002 and featured 81 skeletal DoF and 96 stiff actuators, each equipped with an incremental encoder to measure the length of the tendon, an optical tendon tension sensor and a current sensor in the motor-driver circuit [91, 92, 174]. The 2nd and 3rd robots developed by JSK, Kotaro and Kojiro, were completed in 2006 and 2007, respectively [48, 90, 93–96, 98–101, 143] (see also Figure 2.4). Both robots included prototypes of an optical joint position sensor JSK robots

			S	Skeleton			Muscu	Muscular System	-		
Robot	Year	DoF	Joint	Joint Type / Sensorized	orized	DoF	Technology	Sensors	sors	Multi-	Stiff/
			Revolute	Universal	Spherical		Q	Length ^a	Tension	Articular	Compliant
Kenta [91, 92, 174]	2002	81	•/-	-/-	•/-	96	Electromagnetic	•	•	I	•/-
Kotaro [90, 93–95, 98, 157]	2006	91	•/-	-/-	•/•	90-120 ^b	Electromagnetic	•	•	I	•/-
Kojiro [48, 96, 99–101, 143]	2007	82	•/-	-/-	•/•	109	Electromagnetic	•	•	I	•/•
Kenzoh [51, 52, 102, 112, 113]	2011	46	•/-	•/-	•/-	80	Electromagnetic	•	•	•	•/•
Kenshiro [2, 3, 72, 89]	2012	77	•/-	•/-	•/-	70-100 ^b	Electromagnetic	•	•	•	•/-
Ecce-1 [46, 80, 168]	2009	41	•/-	-/-	•/-	44	Electromagnetic	0	I	•	•/•
Ecce-2 [80][168]	2010	103	•/-	-/-	•/-	44	Electromagnetic	0	I	•	•/•
Ecce-3 [168]	2011	44	•/-	-/-	•/-	40	Electromagnetic	0	•	•	-/•
Ecce-4 [168]	2012	43	•/•	-/-	•/-	52	Electromagnetic	0	•	•	-/•
Anthrob [169]	2012	18	•/•	-/-	•/-	13	Electromagnetic	0	•	•	-/•
Roboy [116]	2013	59 ^c	•/•	-/-	•/-	48	Electromagnetic	•	•	•	-/•
BioBiped1 [84, 123, 124, 133]	2011	8	•/•	-/-	-/-	4	Electromagnetic	0	I	•	-/•
Mowgli [107]	2007	6	•/•	-/-	-/-	6	Pneumatic	I	•	•	-/•
Quadruped [110, 170, 171]	2011	10	•/•	-/-	-/-	10	Pneumatic	I	•	•	-/•
Athlete [105, 106, 108, 109]	2010	6	•/•	-/-	-/-	14	Pneumatic	I	•	•	-/•
Pneuborn-7II [103, 104]	2009	26	•/-	-/-	•/-	19	Pneumatic	I	I	•	-/•
Pneuborn-13 [49, 104]	2009	21	•/-	-/-	•/-	18	Pneumatic	I	I	•	-/•

Table 2.1: Overview of important musculoskeletal robots.

^b additional muscles can be added if required ^c each hand adds an additional 15 DoF

14

for the spherical 3-DoF joints used in the shoulder and hip [157]. Moreover, Kojiro was the first robot built by JSK that featured nonlinear spring units to provide a passive tendon compliance similar to human muscles [96]. More recently, JSK presented two additional musculoskeletal robots. Kenzoh, which was completed in 2011, is a torso with 46 skeletal DoF that included a revised, non-linear spring unit as well as bi-articular actuators (actuators with tendons that span two joints) [51, 52, 102, 112, 113]. In humans, such bi-articular *muscles* are for instance found in the leg and have been reported to contribute to the coordination of joint synchronization [39, 160]. The most recent robot developed by JSK—Kenshiro—is still under construction [2, 3, 72, 89]. It is the most complex Japanese musculoskeletal robot so far and it features all improvements that have been made during one decade of musculoskeletal robot development.

The only musculoskeletal robots worldwide that can compete with the complexity of the JSK robots have been developed by the Eccerobot project consortium between 2009 and 2012. These robots, which are called Ecce-1 through Ecce-4, are all hand-fabricated humanoid torsos (see Figure 2.4 and [168]). The material used for the bones of the robot is a caprolactone polymer which can be easily hand-molded at a temperature of 60 °C. Similar to the latest JSK robots, the Ecces use tendons that are equipped with non-linear series elastic elements to imitate the compliant properties of biological muscles. Ecce-1, which is essentially a refurbished and sensorized version of the Cronos robot, developed as part of a project on machine consciousness at the University of Essex [46], features 44 tendon-driven actuators as well as two different types of shoulder designs. Ecce-2 and Ecce-3 completed in 2010 and 2011, respectively, aimed at replicating the complex skeletal structure of the human spine. However, Ecce-2 does not include any tendon tension sensors, which prohibits the development of state-of-the-art force or torque control algorithms. This limitation was attempted to be eliminated in Ecce-3, but unfortunately with only limited success as the tailored tension sensors proved to be not reliable. Therefore, industrial load cells were used in the final robot developed by the Eccerobot project. However, due to the bulkiness of these sensors, not all actuators could be equipped with tendon tension sensing due to space limitations. Furthermore, none of the robots developed by the Eccerobot project comprises joint position sensors—a prerequisite for classical feedback control algorithms. Therefore, the robot Anthrob was conceptualized towards the end of the Eccerobot project. It features only four skeletal DoF (a spherical shoulder and revolute elbow joint) but provides all required sensor modalities for the development of control algorithms. This robot is presented in detail in Chapter 6.

The Eccerobots

16 BACKGROUND

Musculoskeletal robots with pneumatic actuators The robots presented so far all featured electromagnetic, tendon-driven actuators that require an additional series elastic element to imitate the compliant properties of biological muscles. In contrast, this is not necessary if pneumatic actuators are used which exhibit a non-linear compliance inherent to the actuation principle. However, pneumatic actuators require large compressors which make them unsuitable for mobile robots and are difficult to control. Therefore, they are only found in few musculoskeletal robots, such as in the bipedal robots Mowgli [107] and Athlete [105, 106, 108, 109]. Both robots use pneumatic actuators of the McKibben type which provide a similar length-load curve as biological muscles [107, 152, 153] and demonstrate jumping as well as soft landing due to the intrinsic compliance of the actuation modules.

2.2 JOINT-SPACE SIMULATION OF FORWARD DYNAMICS

Analytical modeling and simulation of robots The dynamics of robots with open-loop kinematics are commonly modeled in joint-space by the following equation:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{\ddot{q}} + \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\dot{q}})\boldsymbol{\dot{q}} + \boldsymbol{g}(\boldsymbol{q}) \tag{2.1}$$

where q, \dot{q} and \ddot{q} (all $\in \mathbb{R}^n$) are the joint positions, velocities and accelerations, respectively, $\tau \in \mathbb{R}^n$ is a vector of joint torques, $M(q) \in \mathbb{R}^{n \times n}$ is a symmetric, positive-definite matrix called the mass or inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ is a matrix that, multiplied with \dot{q} results in an $n \times 1$ vector of centrifugal and Coriolis terms, $g(q) \in \mathbb{R}^n$ is a vector of gravity terms and n is the number of DoF [16, 141, 142]. This equation, which is typically derived by either the Lagrangian or the Newton-Euler formulation, relates the joint accelerations \ddot{q} to joint torques τ . Therefore, in its presented form, it is typically used for robot control where the task is to compute the joint torques required for desired joint accelerations—the *inverse dynamics*. However, if solved for \ddot{q} , Equation 2.1 is equal to:

$$\ddot{q} = M^{-1}(q) [\tau - C(q, \dot{q})\dot{q} - g(q)]$$
(2.2)

In this form, Equation 2.1 can be used for the simulation of the *for-ward dynamics* by employing standard numerical integration methods to compute the joint velocities and positions from the accelerations induced by the applied joint torques [16, 30]. Note that Equation 2.2 requires the mass matrix to be invertible which is the case as *M* is a symmetric, positive-definite matrix [30, 97].

Analytical modeling and simulation of tendon-driven robots If the task is now to simulate the dynamics of tendon-driven robots we can still rely on Equation 2.2 but we have to account for the tendon-driven actuation by finding a mapping *h* between the tendon-forces $f \in \mathbb{R}^m$ and the joint-torques τ :

$$\boldsymbol{\tau} = h(\boldsymbol{q}, \boldsymbol{f}) \qquad h: \mathbb{R}^{n \times m} \to \mathbb{R}^n$$
(2.3)

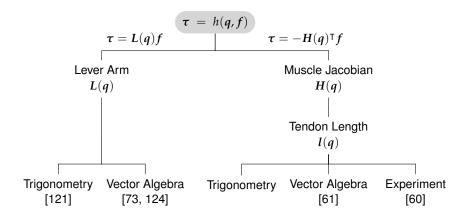


Figure 2.5: Overview of approaches for computing joint torques τ from actuator forces f. The joint torques can be computed from either the lever arm matrix L(q)or from the muscle Jacobian H(q). Both quantities can be derived by either trigonometry or vector algebra while the muscle Jacobian can also be derived experimentally.

where *m* is the number of tendon-driven actuators. For tendon-driven robots with PILAs, *h* is independent of *q* and is equal to:

$$\tau = L f \tag{2.4}$$

where $L \in \mathbb{R}^{n \times m}$ is a matrix defining the lever arms of the actuators. Moreover, if each joint is actuated by a single actuator—as in robots with a revolving tendon routing—L is a diagonal matrix. However, for tendon-driven robots with PDLAs, such as musculoskeletal robots, h(q, f) is not independent of q anymore. In that case, h(q, f) can be computed via two approaches (see also Figure 2.5): (i) via the posture dependent lever arm matrix L(q), reminiscent of the PILA case, and (ii) via the *muscle Jacobian* H(q). Both approaches are described in detail in the following paragraphs.

If the joint axis of rotation and the line of action of the tendon are perpendicular, L(q)—the posture dependent lever arm matrix—can be derived from trigonometric relations. An example is presented in Figure 2.6 where a revolute joint is actuated by a single tendon-driven actuator with a point-to-point tendon routing and a posture-dependent lever arm. Here, the lever arm l_A is a nonlinear function of the angle θ which in turn depends on the joint position ϕ (see Figure 2.6). While this trigonometric approach is feasible for small scale setups, such as the antagonistically actuated revolute joint modeled by Potkonjak et al. [121], it is normally not viable for musculoskeletal robots. First of all, musculoskeletal robots typically feature many actuators and the manual derivation of the lever arm function for each actuator would be a very tedious and error-prone task. Furthermore, if complex joint types with more than one joint axis are used, such as spherical joints, the joint axes and the tendon line of action can never be perpendicular

Trigonometric *derivation of* L(q)

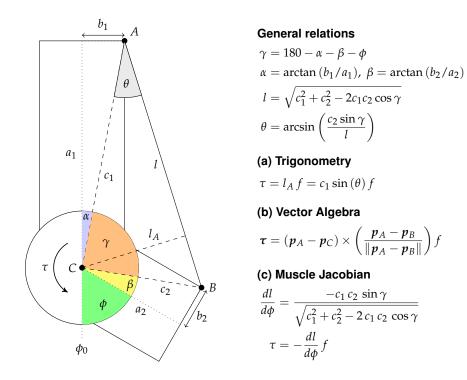


Figure 2.6: Examples of analytical modeling approaches of musculoskeletal robots.

for all joint positions, rendering the trigonometric approach useless. In these cases, L(q) can be computed via vector algebra.

Vector algebraic derivation of L(q)

From classical mechanics it is known that the 3-dimensional torque vector $\boldsymbol{\tau}$ with respect to a reference point is equal to [40]:

$$\boldsymbol{\tau} = \boldsymbol{p} \times \boldsymbol{f} \tag{2.5}$$

where $p \in \mathbb{R}^3$ is the position vector (a vector from the reference point to any point on the line of action of the force) and $f \in \mathbb{R}^3$ is the force vector pointing in the direction of the line of action of the force. This relation can be used to efficiently compute the elements of the lever arm matrix L(q). The position vector is given by a vector from either the joint axis in the case of a revolute joint or from the center of rotation of a spherical joint to one of the tendon attachment points. The force vector can be more complicated to compute-particularly in the case of multi-articular actuators and tendon wrapping on the kinematic structure. However, in the simple case of non-colliding tendons, as in [121, 124] or in Figure 2.6, f is equal to a normalized vector between the two attachment points of the tendon and scaled by the force magnitude f, which in turn depends on the used tendon model. Note that the joint axis and the computed torque vector auhave to be expressed in an identical reference coordinate system, otherwise erroneous torques will be applied. Furthermore, in the case of a revolute or universal joint, the 3-dimensional torque vector au has to

be converted into a scalar representing the magnitude of the torque around a single rotation axis. This *conversion* can easily be achieved by multiplying τ with a unit vector e in the direction of the joint axis. In summary, this yields the following equation for computing the magnitude of the torque τ around the joint axis resulting from the tendon force f:

$$\tau = \left(\boldsymbol{p} \times \hat{f}\right) \cdot \boldsymbol{e} f \tag{2.6}$$

where the term $(p \times \hat{f}) \cdot e$ is the lever arm of the actuator with respect to the joint axis. Therefore, we can use Equation 2.6 to compute the elements of L(q) equal to:

$$\boldsymbol{L}(\boldsymbol{q}) = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{bmatrix}$$

with

 $a_{i,j} = \begin{cases} \left[\boldsymbol{p}_{i,j}(\boldsymbol{q}) \times \hat{f}_j(\boldsymbol{q}) \right] \cdot \boldsymbol{e}_i & \text{if joint axis } i \text{ is affected by actuator } j \\ 0 & \text{otherwise} \end{cases}$

This vector algebraic approach for deriving L(q) is also shown for the simple revolute joint example in Figure 2.6b. Here, the vector p is equal to the vector from the joint center C to the tendon attachment Aand the line of action of the force \hat{f} is given by the normalized vector from the attachment point B to A. As the torque vector and the joint axis are linearly dependent, i. e. parallel, no multiplication with a joint axis unit vector is necessary and we can compute the magnitude of the torque simply by taking the vector norm. From this example it can also easily be shown that the vector algebraic approach yields the same result as the trigonometric approach. From vector algebra it is known that the magnitude of the cross product of two vectors is equal to the product of the magnitudes of the two vectors and the sine of the angle θ between the two vectors. Therefore, the magnitude τ of the torque vector τ is given by:

$$\tau = \|\boldsymbol{\tau}\| = \|\boldsymbol{p}_A - \boldsymbol{p}_C\| \|\hat{f}\| \sin\left(\theta\right) f = c_1 \sin\left(\theta\right) f$$
(2.7)

which is equal to the trigonometric result (see Figure 2.6a). This approach, which has for instance been used by Radkhah et al. to compute the torques induced by the uni- and bi-articular tendon-driven actuators of the BioBiped1 robot [124], is very versatile and can easily be scaled to simulate robots with many active DoF. However, computing the normalized tendon force vector \hat{f} can be particularly difficult for musculoskeletal robots where muscle wrapping can occur. In these cases, the muscle Jacobian can be used.

The muscle Jacobian H(q)

The muscle Jacobian $H(q) \in \mathbb{R}^{m \times n}$ relates tendon length changes to joint position changes and is defined as the partial derivative of the tendon length vector $l \in \mathbb{R}^m$ with respect to the joint positions q [19, 175]:

$$H(q) = \frac{\partial l}{\partial q} = \begin{bmatrix} \frac{\partial l_1}{\partial q_1} & \frac{\partial l_1}{\partial q_2} & \cdots & \frac{\partial l_1}{\partial q_n} \\ \frac{\partial l_2}{\partial q_1} & \frac{\partial l_2}{\partial q_2} & \cdots & \frac{\partial l_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial l_m}{\partial q_1} & \frac{\partial l_m}{\partial q_2} & \cdots & \frac{\partial l_m}{\partial q_n} \end{bmatrix}$$
(2.8)

From the principle of virtual work it can be subsequently shown that [19]¹:

$$\tau = -H^{\mathsf{T}}(q)f \tag{2.9}$$

Derivation of the muscle Jacobian

One approach to compute H(q) would be the trigonometric derivation of a tendon length model and its subsequent differentiation as shown in Figure 2.6c. Here, the tendon length *l* is given by the law of cosines:

$$l = \sqrt{c_1^2 + c_2^2 - 2c_1c_2\cos\left(180 - \alpha - \beta - \phi\right)}$$
(2.10)

Differentiation with respect to ϕ yields:

$$\frac{dl}{d\phi} = \frac{-c_1 c_2 \sin\left(180 - \alpha - \beta - \phi\right)}{\sqrt{c_1^2 + c_2^2 - 2c_1 c_2 \cos\left(180 - \alpha - \beta - \phi\right)}}$$
(2.11)

$$= \frac{-c_1 c_2 \sin (180 - \alpha - \beta - \phi)}{l}$$
(2.12)

which can be shown to be identical to the previously computed lever arms by applying the law of sines to derive a formula that relates θ and the joint angle ϕ . However, this approach yields a model of similar complexity as the trigonometrically derived lever arms shown previously and hence this approach is also normally not feasible for musculoskeletal robots. Another way of computing the muscle Jacobian is approximating it through finite differentiation of the tendon length [140]. Here, small perturbations Δq are made for each DoF and the associated tendon length change Δl is computed for all *m* tendons, e.g. via a vector algebraic or trigonometric model. By subsequent computation of the difference quotient $\Delta l/\Delta q$ the muscle Jacobian can be approximated. However, the small perturbations Δq have to be made for each joint individually while all other joint positions have to be kept constant. Therefore, this approach is time consuming and cannot be used for real-time simulations. In general, if

¹ the negative sign results from the convention that a tendon shortening is associated with a positive force

real-time constraints are considered, the muscle Jacobian has to be computed a priori. One approach that has proven to be feasible has been first evaluated in simulations by Schmaler [132] and was later adapted by Jäntsch et al. for the acquisition of the muscle Jacobian of the Anthrob robot [60, 61]. Here, an Artificial Neural Network (ANN) has been used to approximate the tendon length of all actuators and the difference quotient was used to compute H(q). However, the generation of the training data for the ANN is not trivial, as samples of the entire joint space of the robot have to be drawn to preclude the need for muscle Jacobian extrapolation by the ANN. Furthermore, the question remains how the muscle Jacobian can be learned by this approach for arbitrary kinematic setups without supervision. In summary, the usage of the muscle Jacobian for the simulation of the forward dynamics of musculoskeletal robots is a promising approach as long as an efficient algorithm for its computation can be found.

2.3 SIMULATION TOOLS FOR MUSCULOSKELETAL ROBOTS

Even though many tools exist for the simulation of mobile or industrial robots, such as Microsoft's Robotics Studio [53] or Webots [87], there is no robotic simulation toolbox available that is capable of simulating the forward dynamics of tendon-driven robots. One of the main reasons is the difficulty of automatically deriving the lever arm matrix L(q) or the muscle Jacobian H(q) for an arbitrary robot configuration as pointed out in the previous section. However, problems associated with the simulation of tendon-driven robots are also encountered in biomechanics. Here, skeleton and muscle models are used to investigate aspects of motion dynamics. A few tools have emerged in this field in recent years, such as OpenSim [21] or Anybody. Whereas OpenSim is an academic, free-of-charge project developed at the Center for Biomedical Computation at Stanford, Anybody is a commercial product. Both tools provide detailed models of the human skeleton and include various human-inspired muscle models, such as the Hill model [43]. However, both tools operate in the joint space which means that the tedious procedure of converting muscle forces into joint torques, as presented in the previous section, has to be applied. Furthermore, the closed-source character of Anybody impedes the inclusion of new muscle models required for the simulation of musculoskeletal robots, such as electromagnetic series elastic actuators. Hence, tailored solutions are typically used by roboticists to simulate the dynamics of a particular musculoskeletal robot [33, 124]. To close this gap, a modular and extensible simulation framework for tendon-driven robots, called CALIPER, was developed as part of this thesis. CALIPER is based on the physics-based simulation approach and includes all muscle models presented in Chapter 4. An overview of the functionality of CALIPER can be found in Appendix E.

SKELETON MODELING

The bones of musculoskeletal robots are typically fabricated either by 3D printing (e.g. Roboy or Anthrob) or by more traditional production techniques using aluminium or other metals (e.g. BioBiped). In general, these bony structures are optimized for weight, size and shape (the bone should resemble the human model), rather than for supporting large payloads like the links of industrial robots. Hence, the resulting skeletons are less stiff than an industrial manipulator. But what does this mean for the simulation of the skeletons? Musculoskeletal robots are very different from industrial manipulators and therefore the precision requirements of these two classes cannot be compared. While for industrial robots a repeatability within the submillimeter domain is common, musculoskeletal robots exhibit positioning errors in the order of centimeters due to the compliant actuators and high joint as well as muscle friction forces [58]. Thus, even if the bones would slightly bend under load, the resulting errors would still be negligible in terms of the overall positioning error. Therefore, similar to industrial robots, the skeletons of musculoskeletal robots can be modeled by a system of articulated rigid bodies.

All existing physics-engines support the simulation of articulated rigid body systems by means of rigid body dynamics and Figure 3.1 shows the four stages typically involved in a single simulation step of a rigid body dynamics pipeline: (i) forward dynamics, (ii) collision detection, (iii) constraint solving and (iv) position update. Describing each stage in detail is out of the scope of this dissertation and the interested reader is referred to the PhD thesis of Kenny Erleben [29], the diploma thesis of Helmut Garstenauer [35] or to the book Game Physics by David Eberly [27]. In the context of this dissertation, however, it is important to outline the differences between the standard joint space forward dynamics approach used for the simulation of standard industrial robots as described previously in Section 2.2 and the body space approach of physics-engines. These differences are summarized in Section 3.1. Furthermore, musculoskeletal robots typically feature revolute and spherical joints. The modeling of these joint types as well as of joint friction in a physics-engine is summarized in Section 3.2.

Skeletons & articulated rigid bodies

Physics-engines & rigid body dynamics

24 SKELETON MODELING

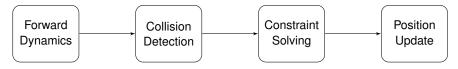


Figure 3.1: Rigid body dynamics pipeline. In physics-engines, the rigid body dynamics pipeline normally comprises four steps. First, the (unconstrained) forward dynamics of each rigid body are computed typically via the Newton-Euler dynamics equations. Then, possible collisions are detected and, if required, contact forces are computed. Subsequently, constraint equations are solved to simulate the dynamics of contacts, collisions and joints. Finally, the world coordinates of the rigid bodies are updated.

3.1 BONES

In a physics-engine, a bone can be represented by a rigid body *B* given by a quadruple of the form:

$$B := ({}^{\mathsf{W}}T_{\mathsf{B}}, m, I, S) \tag{3.1}$$

where ${}^{W}T_{B} \in \mathbb{R}^{4 \times 4}$ is a transformation matrix describing the position and orientation of the center of mass of the body with respect to a world coordinate frame $W, m \in \mathbb{R}^{+}$ is the mass of the body, $I \in \mathbb{R}^{3 \times 3}$ is the inertia tensor of *B* and *S* is a set of optional collision shapes.

Forward dynamics The first stage of a physics-engine simulation pipeline is normally the computation of the forward dynamics (i. e. the prediction of the unconstrained motion of the rigid bodies). Here, the linear a and angular accelerations α of the bodies are computed via Newton's 2nd law of motion and Euler's equation:

$$a = f/m \tag{3.2}$$

$$\boldsymbol{\alpha} = \boldsymbol{I}^{-1}[\boldsymbol{\tau} - \boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega})] \tag{3.3}$$

where *f* and τ are 3D force and torque vectors representing the lumped effect of the external forces (e.g. gravity or muscle forces) and ω is the angular velocity of the body. The term $\omega \times (I\omega)$ in Equation 3.3, which is called the Coriolis term, is sometimes neglected in physics-engines to improve the performance and stability of the simulations.

Bones & muscle forces of musculoskeletal robots apply forces to the bones of the skeleton. These forces normally do not act on the center of mass of the rigid bodies directly but on some point on the body surface. Hence these forces will result in a linear and angular acceleration of the body. Let us consider a muscle force f_M that is applied to body B at point p, where p is given relative to the center of mass. Then the

resulting force f and torque τ acting on the body are given by the following two equations:

$$f = f_M \tag{3.4}$$

$$\boldsymbol{\tau} = \boldsymbol{p} \times \boldsymbol{f}_M \tag{3.5}$$

From the previous paragraph and Equation 3.1, the major difference between the classical robot simulation approach presented in Section 2.2 and the physics-based approach already becomes apparent. While traditional robot simulations operate in the *joint space* (i. e. the joint positions are known and not the body transforms), physicsengines work in the *body space* with no explicit representation of the joint positions. This difference, which might appear to be minor at first, has a major impact on the simulation of musculoskeletal robots. As shown in detail in Section 2.2, the simulation of musculoskeletal robots in the joint-space domain requires the muscle forces to be converted into joint torques. This intermediate step, which can be tedious for more complex setups, is not required if a body space approach is employed. Here, the muscle forces can be applied directly to the rigid bodies of the skeleton, which significantly simplifies the implementation of muscle models as shown in Chapter 4.

Physics-engines normally include algorithms for collision detection and handling. These collision detection algorithms can be used for simulating skeleton self-collisions as well as for skeleton vs. environment collisions—as for instance required for the simulation of handling or interaction scenarios. Collision detection is performed in two steps, called *broadphase* and *narrowphase* collision detection. While in the broadphase, the axis-aligned bounding boxes (AABBs) of the collision shapes are used to determine whether "bounding volumes overlap" [158], the exact contact forces are computed in the narrowphase step. As collision shapes, simple primitives, such as boxes or cylinders, but also arbitrary convex and concave meshes are normally supported. The details, however, depend on the used physics-engine.

3.2 JOINTS

As physics-engines operate in the body-space, simulating joints is more complex than in traditional joint-space approaches. But again, a detailed overview of joint modeling in physics-engines is out-ofscope of this dissertation and the interested reader is referred to [27, 29, 35, 158, 164]. Thus, in the following paragraphs, we will limit ourselves to the basic principles of joint simulations in physicsengines and outline how these algorithms can be used to simulate dry and viscous joint friction. Body space vs. joint space

Collision detection

26 SKELETON MODELING

- Joints & constraints In physics-engines, joints are represented by a set of holonomic constraint equations (i. e. equality constraints that only depend on time and on the position coordinates of the involved body pair) [29, 35]. These constraint equations are either solved sequentially or simultaneously to compute the constraint forces that ensure that the constraint equations are satisfied. Typically, this is done by finding the *best* solution in an iterative loop. As real-time performance is important for physics-engines, the number of iterations of this loop is normally limited and hence, the solution found might be non-optimal leading to simulation artifacts (i. e. breaking constraints). This can be avoided by either reducing the size of the simulation time-step or by increasing the number of solver iterations.
 - *Joint types* Musculoskeletal robots typically feature revolute and spherical joints. Whereas the revolute joint removes 5 DoF (3 linear and 2 angular), the spherical joint only constrains the linear motion of the bodies to be zero, leaving the 3 angular DoF unconstrained. Both of these joint types are supported by popular physics-engines (see [8]). An extensive overview of joint types and joint modeling in general can be found in the dissertation of Kenny Erleben [29].
 - *Joint limits* Typically the movement range of joints is not infinite. For instance, the movement range of the human elbow joint is approximately 145° [119]. These joint *limits* are simulated in physics-engines similar to contact constraints [29, 158]. However, instead of assuming a hard limit, soft limits are used to stabilize the simulation in the vicinity of the limit angles [158].
 - In computer animations and physics-based simulations of traditional Joint motors & friction robots with joint-actuators, it is often required to drive a joint to a reference position. In physics-engines this can be achieved by the use of joint motors, which utilize the error reduction term of constraint equations to invoke a reference joint velocity [29]. In the context of musculoskeletal robots, however, joint motors are not relevant to induce motions-this is done by the muscular system-but to resist motions by modeling joint friction. If we focus for now on rotatory joints, then joint motors are typically parametrized by a reference joint velocity ω_{motor} and by a maximum torque τ_{max}^{1} . By setting $\omega_{\text{motor}} = 0$ and τ_{max} to a non-zero value corresponding to the joint friction torque, the constraint solver will apply any torque in the range $-\tau_{\max} \leq \tau \leq \tau_{\max}$ in order to reach ω_{motor} . This principle can be used to efficiently simulate joint friction.

¹ in the case of linear joints, ω and τ_{max} would be substituted by the linear velocity vector v and by a force vector f_{max}

Three types of friction forces contribute to joint friction: (i) static friction when the norm of the joint velocity $\|\omega_{\text{joint}}\| = 0$, (ii) kinetic friction when $\|\boldsymbol{\omega}_{\text{joint}}\| \neq 0$ and (iii) viscous friction forces which are proportional to $\|\omega_{\text{joint}}\|$. The lumped effect of all three friction types can be modeled by a single friction model, called Stribeck friction curve. This friction model, which was developed by the German engineer Richard Stribeck in 1903 [146], models the transition from static to kinetic friction by a continuous function but exhibits still a discontinuity if the relative velocity of the two objects in contact is zero. Hence, to increase the numerical stability of simulations, the friction force is normally approximated via a linear or sigmoid function in the zero velocity vicinity [1]. This approach is not required if joint motors are used as the iterative algorithm of the constraint solver already guarantees that the joint motor applies any torque between 0 and τ_{max} if $\|\boldsymbol{\omega}_{\text{joint}}\| = 0$ and $\boldsymbol{\tau}_{\max}$ if $\|\boldsymbol{\omega}_{\text{joint}}\| \neq 0$ [29]. Hence, the maximum motor torque of a revolute joint with scalar joint velocity ω is given by the Stribeck equation and equal to:

$$\tau_{\max} = \tau_{\text{Stribeck}} = \tau_k + (\tau_s - \tau_k) e^{-|\omega|/v_s} + k_v \,\omega \tag{3.6}$$

where τ_s and τ_k are static and kinetic friction torques, k_v is the viscous friction coefficient and v_s is the Stribeck velocity. In the case of a spherical joint, the joint velocity and the joint motor torque are vectors and the maximum joint motor torque is equal to:

$$\boldsymbol{\tau}_{\max} = \begin{cases} \boldsymbol{\xi} \begin{pmatrix} \tau_s \\ \tau_s \\ \tau_s \end{pmatrix} & \text{if } \|\boldsymbol{\omega}\| = 0 \text{ with } 1/\sqrt{3} \le \boldsymbol{\xi} \le 1 \\ \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \boldsymbol{\tau}_{\text{Stribeck}} & \text{if } \|\boldsymbol{\omega}\| \neq 0 \end{cases}$$
(3.7)

Thus the motor torque is distributed onto the three joint axis depending on the relative joint velocity if $\|\omega\| \neq 0$ and equally distributed otherwise. The factor ξ would be equal to 1 if the joint movement would be limited to a single axis and $1/\sqrt{3}$ if all three joint axis would be involved. Hence, assuming $\xi = 1/\sqrt{3}$ would apply less friction if only one joint axis would be involved and setting $\xi = 1$ would result in too high friction torques if multiple axis are involved. Hence a good compromise is to use the arithmetic mean, equal to:

$$\xi = \frac{1 + 1/\sqrt{3}}{2} \tag{3.8}$$

Due to the velocity dependence of the friction torque, the maximum motor torque has to be recomputed and reset for each joint during every simulation step. Joint friction modeling

28 SKELETON MODELING

Joint motors vs. direct torque application Instead of using joint motors for simulating joint friction, the friction torque could be applied directly to the rigid bodies of the joint. This approach, however, has two major drawbacks: (i) the static friction would need to be approximated by a linear or sigmoid function and (ii) the simulation stability would be affected negatively as the friction torques would not be taken into account by the constraint solver. Hence, when supported by the physics-engine, joint motors should be favored for modeling joint friction.

4

MUSCLE MODELING

For the class of tendon-driven robots with Posture-Independent Lever Arms (PILAs) and stiff tendons, modeling the muscle dynamics is essentially equal to modeling the actuator dynamics. This is not the case for musculoskeletal robots. Here, the posture-dependent muscle path has to be taken into account to capture the dynamics of the musculo-skeletal interactions.

In this chapter, a physics-based muscle dynamics model is presented. The model, which is outlined in Figure 4.1, consists of: (i) an actuator dynamics model, that converts an actuator input *u* into a tendon length $l_{T_{t}}$ (ii) a muscle kinematics model which computes the current muscle length l_M and the lines of action of the muscle \hat{F} from the robot posture q, and (iii) a Series Elastic Element (SEE) dynamics model which computes the muscle force f used for scaling the lines of action and as load input for the actuator model. While this model defines a generic framework for the physics-based simulation of muscle dynamics, it was evaluated by implementing models of the Anthrob muscles in Chapter 7. Therefore, without any loss of generality, both the actuator dynamics model presented in Section 4.1 and the SEE dynamics model introduced in Section 4.3 are influenced by the Anthrob muscles and could be easily replaced to fit any other type of robot. The main reason is that even though the actuator dynamics and the SEE dynamics influence the muscle dynamics, the major factor in musculoskeletal robots is the muscle path determined by the muscle kinematics. Here, models of increasing complexity are presented in Section 4.2, ranging from the assumption of straight-line connections between the muscle origin and insertion to the computation of the muscle path on arbitrary meshes. Moreover, how to add friction to the muscle models is described in Section 4.4. Finally, an efficient algorithm for computing the muscle dynamics is proposed in Section 4.5 and a discussion of model limitations and possible improvements is given in Section 4.6.

Physics-based muscle dynamics model

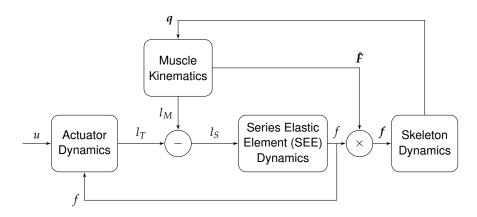


Figure 4.1: Muscle dynamics model. An actuator input *u* results in the tendon length l_T . Subtracting l_T from the muscle length l_M , which in turn depends on the current robot posture *q*, yields the expansion l_S of the SEE. This expansion defines a force *f* which is applied to the skeleton by scalar multiplication with the lines of action of the muscle \hat{F} (also derived from *q*). Finally, the force *f* is fed back to the actuator dynamics model as load input.

4.1 ACTUATOR DYNAMICS

As shown in Figure 4.1, the role of the actuator in modeling the muscle dynamics is to convert an actuator input *u* into a length of the tendon l_T . Different types of actuators, such as pneumatic or electromagnetic actuators, are used in existing musculoskeletal robots for that purpose (see Table 2.1). However, the most common type found are electromagnetic actuators comprising Direct Current (DC) motors. The reasons for that are manifold. First of all, a large variety of DC motors with different functional (e.g. power, speed, size or weight) and economical characteristics (e.g. price) exist on the market from which the robot developer can select the optimal motor for a specific robot and/or use case. Secondly, highly sophisticated position, velocity and torque control methods are available for DC motors which simplifies the high-level joint or operational space control of robots. Finally, whereas pneumatic actuators rely on bulky compressors, ordinary power supplies or even batteries can be used for the operation of DC motors which facilitates the development of autonomous musculoskeletal robots. Two types of DC motors are typically found in musculoskeletal robots: (i) brushed and (ii) brushless DC motors. Brushless motors, on the one hand, provide a higher power-to-weight ratio than brushed motors which reduces the overall weight of the robot as smaller motors can be selected. However, brushless motors also require more complex control electronics and higher gear reduction ratios than brushed motors. Brushed motors, on the other hand, are typically cheaper than brushless motors, a fact which is particularly important in the context of musculoskeletal robots where 40 and more actuators are not uncommon (see Table 2.1). Furthermore, the developed musculoskeletal robot Anthrob uses brushed DC mo-

Electrical subsystem of the DC motor

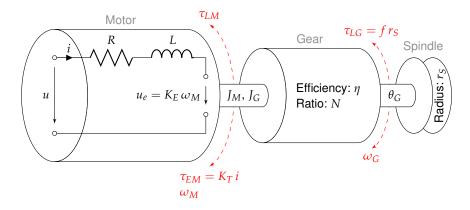


Figure 4.2: Model of an electromagnetic actuator. The actuator consists of a brushed DC motor, which converts the input voltage *u* into an angular velocity and output torque of the motor shaft (ω_M and τ_{EM} , respectively), a gear with efficiency η and ratio *N* that transforms the applied motor torque and velocity, as well as a spindle with radius r_S that translates the angular velocity of the gear output shaft ω_G into a linear velocity of the tendon v_T and the muscle force *f* into the load torque τ_{LG} .

tors. Hence, we focus on the model of a brushed DC motor within this chapter as it is required in Chapter 7 for deriving the physicsbased Anthrob model. Similar models for brushless DC motors or McKibben muscles can be found in [83, 117] and [152], respectively.

4.1.1 Brushed DC Motor Model

The electric equations of a brushed DC motor with permanent magnets can be derived by considering the equivalent circuit diagram shown in Figure 4.2. Applying Kirchhoff's voltage law yields the following first-order linear differential equation [15, 161]:

$$u = Ri + L\frac{di}{dt} + u_e \tag{4.1}$$

where u is the input voltage, R is the armature resistance, L the armature inductance, i the electric current and u_e the Back Electromotive Force (BEMF), which opposes the input voltage in normal operation. The BEMF is equal to:

$$u_e = K_E \,\omega_M \tag{4.2}$$

where ω_M is the angular velocity of the motor output shaft and K_E the BEMF constant. The electromagnetic torque τ_{EM} generated by the motor is linearly dependent on the current and given by:

$$\tau_{EM} = K_T i \tag{4.3}$$

where K_T is the torque constant of the motor¹.

¹ the BEMF and the torque constant already include the constant magnetic flux

32 MUSCLE MODELING

Mechanical subsystem of the DC motor The mechanical equations of a brushed DC motor can be derived by considering the torques acting on the motor shaft and applying Newton's second law of motion (see Figure 4.2). This yields the following equation for the motor shaft acceleration $d\omega_M/dt$ [161]:

$$J_M \frac{d\omega_M}{dt} = \tau_{EM} - \tau_{LM} \tag{4.4}$$

where J_M is the moment of inertia of all moving parts of the motor (rotor, shaft, etc.) and τ_{LM} is the load torque applied to the motor.

Gear model In order to reduce the speed of the motor output shaft and to increase the generated electromagnetic torque, the DC motor is normally equipped with a gear (see Figure 4.2). The governing equations of the speed and torque transformations of an ideal gear (no power losses) with gear ratio *N* are equal to:

$$\omega_i = N \,\omega_o \qquad \tau_i = \frac{\tau_o}{N} \tag{4.5}$$

where τ_i/τ_o and ω_i/ω_o are the input/output torques and angular velocities of the gear, respectively. Thus, by combining the ideal gear and the mechanical subsystem of the brushed DC motor model, we obtain the equation for the angular acceleration of the gear output shaft $d\omega_G/dt$ for the motor/gear assembly:

$$(J_M + J_G)N^2 \frac{d\omega_G}{dt} = N\tau_{EM} - \tau_{LG}$$
(4.6)

where J_G is the gear moment of inertia given at the input shaft and τ_{LG} is the load torque applied to the output shaft.

Gear efficiency However, gears are always subject to power losses. Typically, these losses are due to bearing and gear teeth friction [114], but depend on a variety of gear parameters, such as the type of gear (e. g. spur or planetary) or the number of stages. A simple way of modeling these power losses is by the gear efficiency η , which is defined as the ratio of the output to the input power of the gear²:

$$\eta = \frac{\tau_o \, \omega_o}{\tau_i \, \omega_i} \tag{4.7}$$

However, whether the motor torque or the gear load torque τ_{LG} take on the role of the gear input torque and hence are affected by η depends on the state of the model. Thus, we have to distinguish between two gear operation modes: (i) the forward (normal) and (ii) the reverse (generator) mode (see also Table 4.1). If ω_G and *i* have the same sign, the gear is in forward mode. In this case, τ_{EM} and the inertia of the moving parts are affected by the efficiency, which results in the following equation:

$$(J_M + J_G)N^2\eta \frac{d\omega_G}{dt} = \eta N\tau_{EM} - \tau_{LG}$$
(4.8)

Table 4.1: Gear operation modes. Two operation modes have to be differentiated: (i) the forward (normal) mode in which τ_{EM} and $J_{tot} = J_M + J_G$ are affected by the gear efficiency and (ii) the reverse (generator) mode in which τ_{LG} is affected by the efficiency. The current gear mode can be determined by comparing the signs of the two state variables ω_G and *i*. Please refer to Equation 4.12 for an explanation of the variable ρ .

Model States		Gear Mode	ρ
ω_G	i	Gear Wiode	
> 0	≥ 0	Forward	η
> 0	< 0	Reverse	$1/\eta$
< 0	≤ 0	Forward	η
< 0	>0	Reverse	$1/\eta$

However, if the signs of ω_G and *i* are opposing, the gear is in reverse mode. In this case, the roles of the input and output torques are interchanged (the motor acts as the load) and the load torque τ_{LG} is affected by the gear efficiency. This yields the following equation for the angular acceleration:

$$(J_M + J_G)N^2 \frac{d\omega_G}{dt} = N\tau_{EM} - \eta\tau_{LG}$$
(4.9)

Typically a spindle with radius r_S is attached to the gear output shaft to coil the tendon (see Figure 4.2). The spindle transforms the angular velocity ω_G into a linear velocity of the tendon v_T and the force f into the load torque τ_{LG} . The governing equations are given by:

$$v_T = \omega_G r_S \tag{4.10}$$

$$\tau_{LG} = f r_S \tag{4.11}$$

To simulate the dynamics of the complete actuator model (DC motor, gear and spindle), a state-space model was derived:

$$\frac{d}{dt} \begin{pmatrix} i \\ \omega_G \end{pmatrix} = \begin{pmatrix} -\frac{R}{L} & -\frac{K_E N}{L} \\ \frac{K_T}{NJ_{\text{tot}}} & 0 \end{pmatrix} \begin{pmatrix} i \\ \omega_G \end{pmatrix} + \begin{pmatrix} \frac{1}{L} & 0 \\ 0 & \frac{-r_S}{\rho N^2 J_{\text{tot}}} \end{pmatrix} \begin{pmatrix} u \\ f \end{pmatrix}$$
(4.12)

with $J_{\text{tot}} = J_M + J_G$

and
$$\rho = \begin{cases} \eta & \text{if } \omega_G i \ge 0 \\ 1/\eta & \text{if } \omega_G i < 0 \end{cases}$$

This state-space model can be numerically integrated to obtain the angular velocity ω_G , the position of the gear output shaft θ_G , the motor current *i* and finally the tendon length l_T .

State-space model of the actuator

Spindle model

² for more complex models, please refer to [114]

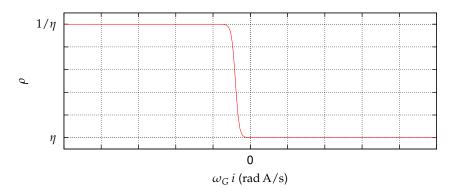


Figure 4.3: Interpolation of gear efficiency ρ for numerical stability. To increase the numerical stability of the simulation, the discontinuous gear efficiency ρ can be approximated by a continuous sigmoid function (here the hyperbolic tangent, see Equation 4.13). When the product of the angular velocity ω_G and the current *i* is positive, the actuator is in forward mode and $\rho = \eta$, otherwise the actuator is in reverse mode and $\rho = 1/\eta$. The function was slightly shifted on the x-axis to ensure that $\rho = \eta$ when the actuator is either not moving or the current is zero (no load).

Discontinuity of function ρ and numerical stability The previously described operation modes of the gear and the resulting discontinuous function ρ can induce numerical instabilities if the integration step-size is too large. This can be avoided by approximating the function ρ with a sigmoid function, such as the logistic curve $(y = 1/[1 + e^{-x}])$ or the hyperbolic tangent $(y = \tanh[x])$. However, in this particular case, the hyperbolic tangent should be preferred over the logistic function as high angular velocities and currents can cause a numeric overflow when the logistic function is used due to its exponential term. The resulting approximation function has the form (see also Figure 4.3):

$$\rho = \eta + \left(\frac{1}{\eta} - \eta\right) \left(\frac{\tanh(-\kappa \,\omega_G \, i - \xi) + 1}{2}\right) \tag{4.13}$$

where the parameters κ and ξ define the steepness and the zero crossing of the approximation, respectively.

4.2 MUSCLE KINEMATICS

Modeling the muscle kinematics and hence the interaction of the muscle with the skeleton is essential for the simulation of the muscle dynamics as they provide the following two substantial quantities for each muscle (see also Figure 4.1): (i) the current muscle length l_M , which is used to compute the expansion of the series elastic element l_S and therefore the force magnitude f, and (ii) the lines of action of the muscle \hat{F} which have to be recomputed for each simulation step due to their dependence on the robot posture q.

Section outline In the following subsections muscle kinematics models of varying complexity will be presented. First, a straight-line kinematics model

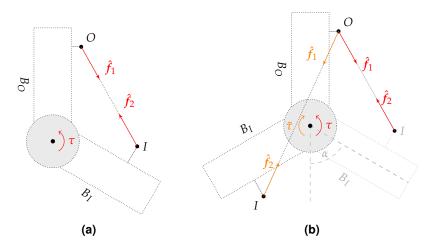


Figure 4.4: Straight-line muscle kinematics model. (a) The most simple way to model a muscle is to assume a straight-line connection between the muscle origin and insertion point (*O* and *I*, respectively). This results in two unit vectors that represent the line of action of the muscle. These units vectors have to be scaled by the force magnitude *f* (derived from the model of the series elastic element) and applied to the rigid bodies B_O and B_I in order to induce the joint torque τ . (b) A straight-line muscle kinematics model can result in erroneous joint torques $\tilde{\tau}$ as shown for the simple case of a revolute joint. If the joint angle α is in the range of $[0^\circ, 180^\circ]$, assuming a straight-line model is correct and the joint torque τ would be applied. However, if $\alpha \in [180^\circ, 360^\circ]$, the assumption of a straight-line model would induce a joint motion in the opposite direction (due to $\tilde{\tau}$).

will be introduced in Section 4.2.1 which is computationally lightweight but suffers from inaccuracy problems in configurations where the muscle would *wrap* around the skeleton. How to approximate such muscle wrappings by adding via-points or basic geometric entities to the muscle path is presented in Section 4.2.2, Section 4.2.3 and Section 4.2.4. Finally, a highly accurate muscle kinematics model, capable of simulating the muscle path on arbitrary meshes, is presented in Section 4.2.5.

4.2.1 Straight-Line Muscle

The most simple way of modeling the muscle kinematics is to assume a straight-line connection between the muscle origin (O, proximal to the actuator) and its insertion point (I, distal to the actuator—see Figure 4.4). This model is computationally lightweight, easy to implement and has therefore been used for almost 40 years for the analysis of biomechanical models [26, 62].

36 MUSCLE MODELING

Muscle lines of action & length In the straight-line muscle kinematics model, the muscle lines of action are given by the following two unit vectors (see also Figure 4.4):

$$\hat{f}_{1} = \frac{p_{I} - p_{O}}{\|p_{I} - p_{O}\|}$$

$$\hat{f}_{2} = -\hat{f}_{1}$$
(4.14)
(4.15)

where
$$p_0$$
 and p_1 are 3D vectors describing the position of the mus-

cle origin *O* and the insertion point *I*, respectively. Furthermore, the muscle length l_M of this model is equal to the Euclidean distance of the origin and insertion point of the muscle:

$$l_{\rm M} = \| \boldsymbol{p}_{\rm O} - \boldsymbol{p}_{\rm I} \| \tag{4.16}$$

Discussion of the straight-line muscle model While being computationally lightweight, there are a number of limitations that have to be considered when using a straight-line muscle kinematics model. The main drawback of this simple model is the fact that the wrapping of muscles around skeletal structures is not captured. While in some setups this might not be required, in the worst-case erroneous joint torques would be applied. This is shown in Figure 4.4b. Here, the straight-line muscle kinematics model would be a good approximation for joint angles α in the range of [0°, 180°], where the muscle lines of action would induce the expected joint torque τ (red lines). However, for joint angles in the range of [180°, 360°], the straight-line muscle kinematics model would fail and even induce the joint torque $\tilde{\tau}$, which would drive the joint in the opposite direction (orange lines). This could be avoided by adding a spherical wrapping surface to the center of rotation of the joint as shown in Section 4.2.3.

4.2.2 Muscle Via-Points

One way to extend the straight-line muscle kinematics model presented in the previous section is to introduce additional waypoints along the muscle path. These waypoints, which are typically called *via-points* [23, 34], can increase the valid range of joint motions and therefore, in some circumstances, yield a better muscle kinematics approximation than the straight-line model.

Muscle lines of action Let M be a totally ordered finite set of n via-points V defining the muscle path from the muscle origin to the insertion point:

$$M := \{V_1 = O, \dots, V_n = I\} \qquad n \ge 2 \tag{4.17}$$

Furthermore, let each via-point *V* be given by a pair comprising the position $p \in \mathbb{R}^3$ of the via-point and its corresponding rigid body *B* (the body to which it is attached):

$$V := (\boldsymbol{p}, B) \tag{4.18}$$

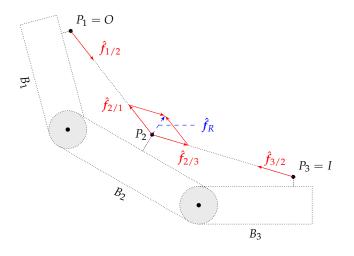


Figure 4.5: Example muscle path with via-point. The straight-line muscle path model presented in Section 4.2.1 can be generalized and extended by defining the muscle path from the muscle origin to its insertion as a finite number of *via-points* V := (p, B), where p is the position of the via-point and B its corresponding rigid body. By pairwise computing the vector difference for all via-points, 2n - 2 unit vectors can be derived that determine the directions in which the muscle force is acting (e.g. $\hat{f}_{1/2}$). Under the assumption that the via-points are frictionless, scaling the unit vectors by the force magnitude f and applying them to the corresponding rigid body B of each via-point V yields automatically the correct reaction forces for the muscle (see P_2 and \hat{f}_R).

Then 2n - 2 muscle lines of action are defined by these *n* via points, two for each pair of consecutive via-points:

$$\hat{f}_{i/i+1} = \frac{p_{i+1} - p_i}{\|p_{i+1} - p_i\|} \qquad 1 \le i \le n-1$$
(4.19)

$$\hat{f}_{i+1/i} = -\hat{f}_{i/i+1} \tag{4.20}$$

where $f_{a/b}$ is the line of action from via-point *a* to via-point *b* (see also Figure 4.5). While these unit vectors determine the directions in which the muscle force is acting, they do not define the magnitude of the force. However, if we assume that the via-points are frictionless, the force magnitude *f* is constant for the entire muscle path and we can linearly scale the lines of action $\hat{f}_{a/b}$ by *f* (how to take friction into account is described in Section 4.4):

$$f_{a/b} = f \, \hat{f}_{a/b} \tag{4.21}$$

The resulting 2n - 2 scaled lines of action $f_{a/b}$ can be finally applied to the corresponding rigid body B_a of via-point a (the details depend on the used physics engine). With this algorithm it is automatically ensured that the correct muscle forces are applied to the rigid bodies, independent of the number of via-points as illustrated in Figure 4.5 for a muscle with one via-point. In this example, it is shown that the direction of the reaction force \hat{f}_R of via-point V_2 is equal to the sum of the two unit forces applied ($\hat{f}_{2/1}$ and $\hat{f}_{2/3}$, respectively).

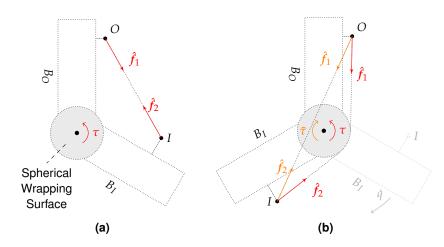


Figure 4.6: Spherical wrapping surface example. (a): Joint position q_1 . In this position, the muscle is collision-free and the lines of action of the muscle \hat{f} can be computed directly from the origin and insertion point (*O* and *I*, respectively). (b): Joint position q_2 acquired from q_1 . In this position, the assumption of a collision-free tendon routing would lead to an erroneous joint torque $\tilde{\tau}$ (orange lines) which would drive the joint in the opposite direction. However, introducing a sphere in the center of rotation of the joint as wrapping surface, yields the desired joint torque τ (red lines).

Muscle length If via-points are considered, the muscle length is given by the sum of the n - 1 Euclidean distances of the n position vectors p_i :

$$l_M = \sum_{i=1}^{n-1} \|\boldsymbol{p}_i - \boldsymbol{p}_{i+1}\|$$
(4.22)

4.2.3 Spherical Wrapping Surfaces

Both the straight-line and the via-point muscle kinematics model are unable to handle the revolute joint setup shown previously in Figure 4.4b. Therefore, in this section, it is now described how to solve this problem by introducing spherical wrapping surfaces into the muscle path as illustrated in Figure 4.6 for the setup used in Figure 4.4b. Here, a spherical wrapping surface is defined in the center of rotation of the joint. This wrapping surface changes the lines of action of the muscle (red lines) as well as the muscle length and induces the desired joint torque τ .

Spherical wrapping surfaces within the muscle path by extending the totally ordered finite set of via-points M (see Section 4.2.2) by 5-tuples of the form $S^S := (p_C, B, r, \chi, \Lambda)$, where p_C is the position of the sphere center, B its corresponding rigid body, r the sphere radius, χ the initial state of the state machine defining the wrapping direction (see below) and Λ the initial value of the wrapping angles greater than 180°:

$$M := \{V_1, S_1^S, V_2, \dots, S_m^S, V_n\} \qquad m \ge 0 \tag{4.23}$$

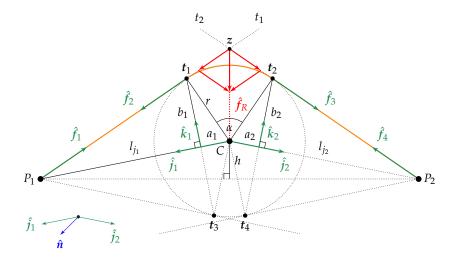


Figure 4.7: Spherical wrapping surface kinematics. Under the assumption that the sphere surface is frictionless, the muscle path along the sphere is equal to the shortest path which lies always within the plane spanned by p_1 , p_c and p_2 . Therefore, the 3D problem can be converted into a 2D problem as shown in the figure. Here, the two muscle via-points (p_1 and p_2 , respectively) define four tangent points (t_1 to t_4) and two muscle paths that differ in length. The selection of the proper muscle path requires knowledge of both the current and the previous postures of the robot (see text). Once a path has been selected, the line of action of the reaction force \hat{f}_R induced by the sphere can be computed efficiently by splitting the muscle path into two straight-line muscles (here from p_1 to t_1 and from t_2 to p_2 , respectively). Finally, the muscle length l_M can be updated by taking the arc length $l_{arc} = \alpha r$ into account.

It should be noted that each spherical wrapping surface S^S has to be enclosed by two via-points (e. g. V_1 and V_2), otherwise the muscle path cannot be computed by the presented algorithm.

Let us consider a muscle segment comprising a preceding via-point V_1 , a spherical wrapping surface S^S and a succeeding via-point V_2 . If we assume that the sphere surface is frictionless, then the muscle path along the sphere is equal to the shortest path. This path always lies within the plane spanned by p_1 , the sphere center p_C and p_2 . Therefore, the 3D problem of computing the muscle kinematics of spherical wrapping surfaces simplifies to a 2D problem as shown in Figure 4.7. Here, the muscle (orange line) induces a reaction force \hat{f}_R with magnitude *f* on the corresponding rigid body *B* of the sphere. Furthermore, the line of action of this reaction force is equal to a straight line through the points z (the point of intersection of the tangents t_1 and t_2) and the sphere center. However, instead of computing \hat{f}_R via z and p_C , it is computationally more efficient to calculate the two tangent points (t_1 and t_2) and to cut the wrapped muscle path into two straight-line muscles (from p_1 to t_1 and from t_2 to p_2 , respectively). Then, the forces acting on the rigid body of the spherical wrapping surface due to the wrapped muscle can be applied by the algorithm presented for the via-points.

Effect of muscle wrapping on its lines of action

40 MUSCLE MODELING

Computation of tangent points

The two via-points that enclose a spherical wrapping surface define four tangent points, which can be efficiently computed via vector algebra and trigonometry as follows (see also Figure 4.7):

 Compute the unit vectors j₁ and j₂ (pointing from p_C to p₁ and to p₂, respectively) as well as the length of these segments (l_{j1} and l_{j2}):

$$\hat{j}_{1} = \frac{p_{1} - p_{C}}{l_{j_{1}}} \qquad l_{j_{1}} = \|p_{1} - p_{C}\|$$
$$\hat{j}_{2} = \frac{p_{2} - p_{C}}{l_{j_{2}}} \qquad l_{j_{2}} = \|p_{2} - p_{C}\|$$

2. Calculate the surface normal \hat{n} of the plane defined by the three coplanar points (p_1 , p_2 and p_C) via the cross product of \hat{j}_1 and \hat{j}_2 :

$$\hat{n} = \hat{j}_1 \times \hat{j}_2$$

3. Compute the vector \hat{k}_1 (perpendicular to \hat{n} and \hat{j}_1) as well as the vector \hat{k}_2 (perpendicular to \hat{n} and \hat{j}_2):

$$\hat{k}_1 = \hat{j}_1 imes \hat{n} \qquad \hat{k}_2 = \hat{n} imes \hat{j}_2$$

4. Compute the length of the segments a_1 and a_2 :

$$a_1 = \frac{r^2}{l_{j_1}}$$
 $a_2 = \frac{r^2}{l_{j_2}}$

5. Compute the length of the segments b_1 and b_2 (e.g. using the Pythagorean theorem):

$$b_1 = \sqrt{r^2 - a_1^2}$$
 $b_2 = \sqrt{r^2 - a_2^2}$

6. Finally, compute the four tangent points:

$$t_1 = p_C + a_1 \hat{j}_1 + b_1 \hat{k}_1 \qquad t_2 = p_C + a_2 \hat{j}_2 + b_2 \hat{k}_2 t_3 = p_C + a_1 \hat{j}_1 - b_1 \hat{k}_1 \qquad t_4 = p_C + a_2 \hat{j}_2 - b_1 \hat{k}_2$$

Four tangent points define two muscle paths The four tangent points define two possible muscle paths: one that wraps in a right-handed (t_3 and t_4) and one that wraps in a left-handed (t_1 and t_2) sense with respect to the normal \hat{n} and the muscle path (from the origin to the insertion). Therefore, the correct tangent point pair has to be selected in order to apply the reaction force f_R to the rigid body of the sphere. Obviously, this selection cannot be made exclusively based on the current posture q(t) of the robot but requires information about the posture during the previous simulation step

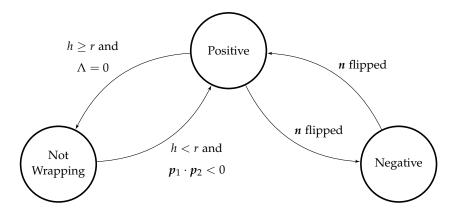


Figure 4.8: Spherical wrapping surface state machine. The four tangent points of a spherical wrapping surface yield two possible muscle paths (see text) and the selection of the proper path depends on the current posture of the robot q(t) and the posture during the preceding simulation step $q(t - \Delta t)$. Therefore, a finite state machine was developed which facilitates the selection of the proper path. This state machine defines three states: (i) *Positive*, which corresponds to a wrapping of the muscle in the right-handed sense with respect to the normal n and the muscle path (from the origin to the insertion), (ii) *Negative* representing a wrapping in the left-handed sense (as in the example of Figure 4.7) and (iii) *Not Wrapping* in which the wrapping surface is inactive and straight-line muscle kinematics between the two enclosing via-points are used. A detailed description of the parameters required for the evaluation of the state transitions is given in the text and in Figure 4.7, respectively.

 $q(t - \Delta t)$. This becomes clear if we consider again the scenario shown in Figure 4.6b. Here, it would be impossible to determine the muscle path if only the joint position q_2 was considered. Therefore, a state machine has been derived that facilitates the selection of the proper muscle path.

The state machine of the spherical wrapping surface model is shown in Figure 4.8. It defines the following three states: (i) Positive, which corresponds to a wrapping in the right-handed sense, (ii) Negative representing a wrapping in the left-handed sense (as in the example of Figure 4.7) and (iii) Not Wrapping in which the wrapping surface is inactive and straight-line muscle kinematics between the two enclosing via-points are used. The initial state χ depends on the initial posture of the robot and has to be user provided. All variables required for the evaluation of the state transitions have been introduced previously, except for the *flipping* of the surface normal n, the height hof the triangle p_1 , p_C and p_2 , and the revolution counter Λ (see next paragraph). The surface normal is considered to have *flipped* when the vector dot product of the normal of the current and previous simulation step is less than 0-i.e. the angle between the two normals is larger than 90°. This happens when the angle between the two vectors j_1 and j_2 becomes larger than 180°, as shown in Figure 4.9a. Here the succeeding via-point moved from P_2 in the previous to \tilde{P}_2 in the current simulation step which caused the normal to *flip*. The height *h*, which is used to determine whether the muscle wraps on the surface

Spherical wrapping surface model state machine

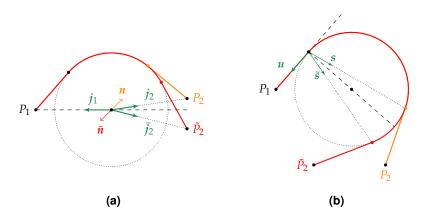


Figure 4.9: Normal *flipping* and revolution counter increment example. (a) The normal $n = j_1 \times j_2$ is considered to have *flipped* when $n \cdot \tilde{n} < 0$ (i. e. the angle between the normal from the previous and current simulation step is larger than 90°). (b) The revolution counter Λ is used to support wrapping angles $\alpha > 180^\circ$. Whether the counter is incremented or decremented depends on the current state of the state machine and the change of sign of the projection of the vector *s* on the vector *u* (see Table 4.2). In the example shown, the state would be equal to *Negative* and Λ would be incremented.

or whether a straight-line muscle model between p_1 and p_2 has to be used, is given by:

$$h = l_{j_1} \sin \left[\arccos \left(-\hat{j}_1 \cdot \frac{p_2 - p_1}{\|p_2 - p_1\|} \right) \right]$$
(4.24)

Multiple wrapping revolutions and the angle α The range of motion of a joint in a musculoskeletal robot is normally less than 360° (in fact, most joints have even smaller movement ranges). Therefore, the maximum angle α between the two tangent points of the muscle path (see Figure 4.7) is typically below 180° and α can be computed via the law of cosines:

$$\alpha = \arccos\left(1 - \frac{t_a t_b}{2 r^2}\right) \tag{4.25}$$

where $\overline{t_a t_b}$ is the Euclidean distance between the two tangent points that define the arc (t_1 and t_2 in Figure 4.7). However, in some special use-cases, wrapping angles of $\alpha > 180^{\circ}$ might be of interest. Therefore, the revolution counter Λ was added to the state machine. It counts multiples of 180° by analyzing the change of sign of the projection of the vector s (the vector from t_a to t_b) onto the vector u (the line joining t_a and p_1 —see Figure 4.9b). However, whether Λ is incremented or decremented also depends on the current state of the state machine. A summary of all four increment/decrement conditions is

Table 4.2: Increment and decrement conditions of the revolution counter. The revolution counter Λ counts multiples of 180° by analyzing the change of sign of the projection of the vector *s* (the vector from t_a to t_b) onto the vector *u* (the line joining t_a and p_1 —see also Figure 4.9b).

State	$ \boldsymbol{u}(t-\Delta t) \cdot \boldsymbol{s}(t-\Delta t)$	$ \boldsymbol{u}(t) \cdot \boldsymbol{s}(t)$	Δ
Positive	< 0	> 0	-1
1 OSITIVE	> 0	< 0	+1
Nagativa	< 0	> 0	+1
Negative	> 0	< 0	-1

given in Table 4.2. If we take the revolution counter into account, the wrapping angle is equal to:

$$\alpha_{\Lambda} = 2 \pi \left\lceil \frac{\Lambda}{2} \right\rceil + \delta$$
(4.26)
with $\delta = \begin{cases} +\alpha & \text{if } \Lambda \mod 2 = 0 \\ -\alpha & \text{otherwise} \end{cases}$

Providing a generic formula for computing the muscle length l_M when multiple spherical wrapping surfaces along the muscle path are defined is difficult due to the dependency of l_M on the number of spheres defined and their current state (e.g. whether the muscle is wrapping or not). However, the length $l_{M'}$ of a muscle segment for which a sphere is defined, as presented in the previous paragraphs, can be easily derived and is equal to:

$$l_{M'} = \begin{cases} \|\boldsymbol{p}_1 - \boldsymbol{t}_a\| + \alpha_{\Lambda} r + \|\boldsymbol{t}_b - \boldsymbol{p}_2\| & \text{if } \chi \neq \text{NotWrapping} \\ \|\boldsymbol{p}_1 - \boldsymbol{p}_2\| & \text{otherwise} \end{cases}$$

$$(4.27)$$

Even though the presented spherical surface model is capable of simulating the kinematics of a muscle wrapping on spherical surfaces, there are some model limitations and simplifications which have to be considered. First of all, the definition of two consecutive spheres along the muscle path is not supported. The reason is, that the algorithm described above for computing the four tangent points on the sphere requires three fixed coordinates (i. e. the sphere center and two muscle via-points). However, in standard musculoskeletal robots this limitation is not problematic, as most muscles are uni-articular with only a single wrapping surface along the muscle path. Furthermore, the sphere surface was assumed to be frictionless, but frictional muscle forces play an important role in the muscle dynamics of musculoskeletal robots [58]. Hence, Section 4.4 introduces how to extend Muscle length

Discussion of the spherical wrapping surface model

the spherical wrapping surface model with friction forces that act along the muscle path. Finally, as the presented algorithm computes the shortest muscle path along the sphere, the muscle can *slide* sideways on the surface if the muscle path is not perpendicular to the axis of rotation of the joint. While this behavior might be desired if the sphere is used in a spherical joint setup, it would result in erroneous force vectors and muscle lengths for a revolute joint. Here, using a cylindrical wrapping surface defined parallel to the joint axis would yield better results.

4.2.4 Cylindrical Wrapping Surfaces

An algorithm for computing the muscle kinematics when cylindrical wrapping surfaces are considered within the muscle path was developed. This algorithm, which is similar to the one presented by Garner and Pandy [34], is essentially an extension of the spherical wrapping surface model presented in the previous section. It assumes a cylinder of infinite height and, in contrast to [34], is capable of simulating the kinematics resulting from multiple muscle wrapping revolutions.

Cylindrical wrapping surfaces within the muscle path Frictionless cylindrical wrapping surfaces can be added to the muscle path by extending the totally ordered finite set of via-points M (see Section 4.2.2) by 5-tuples of the form $S^C := ({}^BT_C, B, r, \chi, \Lambda)$, where BT_C is a transformation matrix describing the position and orientation of the cylinder relative to the rigid body B, r the cylinder radius, χ the initial state of the state machine defining the wrapping direction and Λ the initial value of the wrapping revolution counter (see Section 4.2.3):

$$M := \{V_1, S_1^C, V_2, \dots, S_m^C, V_n\} \qquad m \ge 0$$
(4.28)

Similar to the spherical wrapping surface described in Section 4.2.3, each cylinder S^{C} has to be enclosed by two via-points (e. g. V_1 and V_2), otherwise the muscle path cannot be computed with the presented algorithm. Furthermore, it should be noted that no cylinder height is included in the model as the cylinder is assumed to have infinite height.

Muscle lines of
actionAs the cylinder surface is assumed to be frictionless, the muscle path
along the cylinder is equal to the shortest Euclidean path from the
preceding to the succeeding via-point (p_1 and p_2 , respectively) across
the cylinder surface. But what is the shortest path across a cylinder?
For the spherical wrapping surface model it was possible to convert
the 3D into a 2D problem which could be solved using vector algebra
and trigonometrical functions. But this was only possible as the short-
est path across a sphere always lies within the plane spanned by p_1 ,
 p_2 and the sphere center p_C . This is not the case for a cylindrical wrap-
ping surface, where the path can cross the surface on any point along

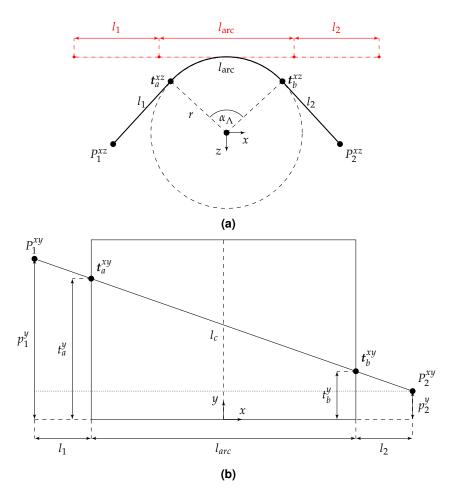


Figure 4.10: Cylindrical wrapping surface model. A cylindrical wrapping surface model was developed to simulate the muscle kinematics if cylindrical obstacles (e.g. bones) are part of the muscle path. (a) Cross-sectional view (x-z plane) of a cylinder. The preceding and succeeding via-points are projected into the x-z plane of the cylinder to compute the tangent points t_a^{xz} and t_b^{xz} using the algorithm presented in Section 4.2.3. (b) *Unwrapped* view of the muscle path. The y-coordinate of the tangent points are subsequently computed by *unwrapping* the path (converting the bent path into a linear path, see also red lines in panel a) and applying similar triangle formulas.

the cylinder. However, the constraint that the muscle path along the cylinder is equal to the shortest Euclidean path still holds if friction is neglected and it can be shown that this path is given by *unwrapping* the cylinder and connecting the preceding and succeeding via-point by a straight line [34]. Therefore, the problem of computing the shortest path across a cylindrical surface can be solved by (i) projecting the via-points into the x-z plane of the cylinder, (ii) computing the x-z coordinates of the tangent points and (iii) back-projecting the tangent points by computing their y-coordinate. In detail, this algorithm proceeds as follows (see also Figure 4.10):

1. First, we compute the projections p_1^{xz} and p_2^{xz} of p_1 and p_2 onto the x-z plane of the cylinder.

- 2. Then we compute the x-z coordinates of the four tangent points $(t_1^{xz} t_4^{xz})$ using the algorithm described in Section 4.2.3.
- 3. These four tangent points define again two muscle paths. We select the proper path (t_a^{xz} and t_b^{xz} , respectively) using the previously introduced state machine (see Section 4.2.3).
- 4. Once these points are selected, we compute the length of the segments *l*₁ and *l*₂ (from *p*₁^{xz} to *t*_a^{xz} and from *t*_b^{xz} to *p*₂^{xz}, respectively) as well as the length of the arc *l*_{arc} in the x-z plane:

$$l_1 = \|\boldsymbol{p}_1^{xz} - \boldsymbol{t}_a^{xz}\| \quad l_2 = \|\boldsymbol{t}_b^{xz} - \boldsymbol{p}_2^{xz}\| \quad l_{\text{arc}} = \alpha_{\Lambda} r$$
(4.29)

5. Now we project the tangent points back into \mathbb{R}^3 by computing their y-coordinate using similar triangle formulas (see Figure 4.10b):

$$t_a^y = p_2^y + \frac{(l_{\rm arc} + l_2)(p_1^y - p_2^y)}{l_1 + l_{\rm arc} + l_2}$$
(4.30)

$$t_b^y = p_2^y + \frac{l_2 \left(p_1^y - p_2^y \right)}{l_1 + l_{\rm arc} + l_2} \tag{4.31}$$

- 6. Finally, we can compute the lines of action of the muscle by assuming two straight-line connections similar to the spherical wrapping surface (from p_1 to t_a and from t_b to p_2 , respectively).
- Muscle lengthSimilar to the spherical wrapping surface, providing a generic for-
mula for computing the muscle length l_M when cylindrical wrapping
surfaces are part of the muscle path is difficult. Therefore, only the
formula for computing the length $l_{M'}$ of a muscle segment for which
a cylinder is defined is provided:

$$l_{M'} = \begin{cases} \|\boldsymbol{p}_1 - \boldsymbol{t}_a\| + l_c + \|\boldsymbol{t}_b - \boldsymbol{p}_2\| & \text{if } \chi \neq \text{NotWrapping} \\ \|\boldsymbol{p}_1 - \boldsymbol{p}_2\| & \text{otherwise} \end{cases}$$

(4.32)

with
$$l_c = \sqrt{(t_a^y - t_b^y)^2 + l_{\rm arc}^2}$$
 (4.33)

Discussion of the cylindrical wrapping surface model The presented cylindrical wrapping surface model is ideal for modeling the kinematics of muscles that wrap around cylindrical bones or for revolute joints where the muscle path is not perpendicular to the joint axis—a setup where the spherical wrapping surfaces would perform poorly. Furthermore, in contrast to the implementation of a cylindrical wrapping surface described by Garner and Pandy [34], the presented algorithm is able to handle multiple revolutions by exploiting the previously described finite state machine (see Section 4.2.3). However, there are also some limitations that have to be discussed which result from the fact that the cylinder is assumed to have infinite height. First, the path might be incorrectly computed to be on the cylinder surface even though it should cross the cylinder edge and second, it is possible that either the preceding or succeeding via-point penetrates the cylinder (particularly if spherical joints are simulated and the cylinder is used to approximate the bone surface). While the first problem has only a minor impact on the simulation results (i. e. a longer muscle and deviations in its lines of action), the second problem makes it impossible to compute the tangent points. However, the presented algorithm could be extended to handle both problems—if required.

4.2.5 Meshes as Wrapping Surfaces

All muscle kinematics models presented so far compute approximations of the exact muscle path based on user-defined via-points and/or wrapping surfaces. While being computationally lightweight, this approach suffers from two problems: (i) defining the via-points and parameterizing wrapping surfaces can be a tedious task if complex musculoskeletal robots are considered and (ii) the computed muscle kinematics (muscle length and lines of action) are only rough approximations of the exact solution if complex bone shapes are considered. Therefore, an algorithm for approximating the muscle path by considering the underlying meshes of the robot skeleton as wrapping surfaces is presented in this section. This approach, which is similar to the algorithm presented by Desailly et al. [23], is particularly useful as the meshes defined in the robot design phase can be directly reused for simulating the muscle kinematics without further adjustments.

Frictionless mesh wrapping surfaces can be added to the muscle path by extending the totally ordered finite set M describing the muscle path from the muscle origin to its insertion by 5-tuples of the form $S^M := ({}^S\hat{\chi}, B, {}^BT_M, {}^MV, U)$, where ${}^S\hat{\chi} \in \mathbb{R}^3$ is a unit vector, given in the frame of reference of the rigid body of the succeeding viapoint and defining the wrapping direction of the muscle for the initial posture of the robot, B is the corresponding rigid body of the mesh, BT_M a transformation matrix defining the position and orientation of the mesh with respect to B, ${}^MV := \{{}^Mv_1, \ldots, {}^Mv_k\}$ is a set of mesh vertices ${}^Mv \in \mathbb{R}^3$, each expressed in the frame of reference of the mesh, and $U := \{u_1, \ldots, u_l\}$ is a set of triangulated mesh facets $u \in \mathbb{N}^3$ indexing MV :

Meshes as wrapping surfaces within the muscle path

$$M := \{V_1, S_1^M, V_2, \dots, S_m^M, V_n\} \qquad m \ge 0$$
(4.34)

Similar to both wrapping surfaces described in the previous sections, each mesh S^M has to be enclosed by two via-points (e.g. V_1 and V_2),

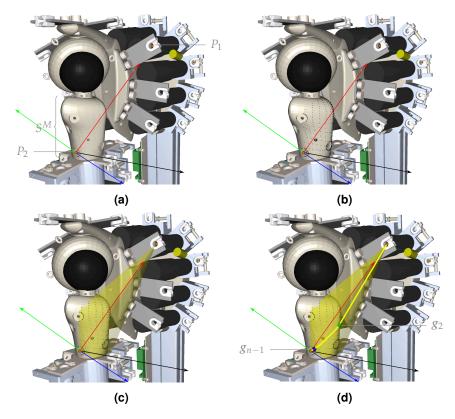


Figure 4.11: Mesh wrapping surface example. The four steps involved in the computation of the geodesic muscle path are shown for the muscle *Anterior Deltoid* of the Anthrob physics-based model (see Chapter 7). In this example, the upper part of the humerus bone was used as wrapping mesh S^M . (a) Step 1: Based on the user-provided wrapping direction vector $\hat{\chi}$ (black arrow), a coordinate frame is computed in the succeeding via-point p_2 . (b) Step 2: The half-space *H* is identified (black dots). (c) The convex envelope of *H*, P_1 and P_2 is computed. (d) The exact geodesic path from P_1 to P_2 on the convex envelope (yellow line) and the path points used for computing the lines of action (g_2 and g_{n-1} , respectively) are found using the algorithm described by Mitchell et al. [88].

otherwise the muscle path cannot be computed by the presented algorithm.

Muscle path If friction is neglected, the muscle path along the mesh is given by the shortest Euclidean path from V_1 to V_2 which does not intersect with S^M . This problem, which is a standard problem in 3D path planning, has been proven to be NP-hard by John Canny and John Reif [12]. However, by computing the convex envelope of both, the mesh S^M and the two enclosing via-points, finding the shortest Euclidean path is identical to computing the shortest geodesic path on a convex polyhedron, which can be solved in polynomial time [78]. In a nutshell, the algorithm for computing the muscle path on a mesh proceeds in four steps, each described in detail in the following paragraphs (see also Figure 4.11): (i) first, a coordinate frame *F* is computed in the succeeding via-point of S^M by taking the user-provided wrapping direction vector ${}^S \hat{\chi}$ into account, (ii) the half-space *H*—the subspace of \mathbb{R}^3

that contains all mesh vertices relevant for the path computation—is determined, (iii) the convex envelope C_E of the half-space and finally (iv) the geodesic path *G* on this envelope is calculated.

In the case of the spherical and cylindrical wrapping surface (see Section 4.2.3 and Section 4.2.4), the initial wrapping direction of the muscle was given by the initial state χ of the state machine (*Positive*, *Negative* or *Not Wrapping*). A similar approach is not possible for mesh surfaces as the path on the mesh cannot be mapped to a 2D problem. Hence, χ was replaced by the unit vector ${}^W \hat{\chi}$ which points into the direction of muscle wrapping and has its origin in the succeeding viapoint (see black vector in Figure 4.11a). Let ${}^W R_S$ be the rotation matrix representing the orientation of the rigid body of the succeeding viapoint with respect to the world coordinate frame, then ${}^W \hat{\chi}$ is equal to:

$${}^{W}\hat{\boldsymbol{\chi}} = {}^{W}\boldsymbol{R}_{S}{}^{S}\hat{\boldsymbol{\chi}} \tag{4.35}$$

Once ${}^{W}\hat{\chi}$ has been computed, we can proceed by defining the coordinate frame *F*. Let ${}^{W}p_1$ and ${}^{W}p_2$ be the origin of the preceding and succeeding via-point of mesh S^M expressed in the world coordinate system, then $F = ({}^{W}\hat{x}, {}^{W}\hat{y}, {}^{W}\hat{z})$ with origin ${}^{W}p_2$ is given by (see Figure 4.11a):

$${}^{W}\hat{\boldsymbol{x}} = \frac{{}^{W}\boldsymbol{p}_{1} - {}^{W}\boldsymbol{p}_{2}}{\|{}^{W}\boldsymbol{p}_{1} - {}^{W}\boldsymbol{p}_{2}\|} \qquad {}^{W}\hat{\boldsymbol{y}} = {}^{W}\hat{\boldsymbol{\chi}} \times {}^{W}\hat{\boldsymbol{x}} \qquad {}^{W}\hat{\boldsymbol{z}} = {}^{W}\hat{\boldsymbol{x}} \times {}^{W}\hat{\boldsymbol{y}}$$

$$(4.36)$$

Half-space

Now that *F* is defined, we can compute the half-space *H* (i. e. the subspace of \mathbb{R}^3 that includes all vertices of the mesh that are relevant for the path computation). Therefore, we first transform all mesh vertices into the world frame *W*. Let ${}^W T_B$ be the transformation matrix of the rigid body *B* with respect to the world coordinate frame and ${}^M v \in {}^M V$ be the coordinates of a mesh vertex expressed in the mesh coordinate frame, then ${}^W v$ is equal to:

$$\begin{bmatrix} {}^{W}\boldsymbol{v} \\ 1 \end{bmatrix} = {}^{W}\boldsymbol{T}_{B}{}^{B}\boldsymbol{T}_{M} \begin{bmatrix} {}^{M}\boldsymbol{v} \\ 1 \end{bmatrix}$$
(4.37)

In the next step we subtract ${}^{W}p_{2}$ from ${}^{W}v$ to obtain ${}^{W}\tilde{v}$ —the mesh vertex coordinates relative to ${}^{W}p_{2}$:

$${}^{\mathrm{W}}\tilde{v} = {}^{\mathrm{W}}v - {}^{\mathrm{W}}p_2 \tag{4.38}$$

Now let ${}^{W}\tilde{V}$ be the ordered set of all mesh vertices expressed in frame *W* and relative to ${}^{W}p_{2}$, then the half-space *H* is given by:

$$H := \{ {}^{\mathsf{W}}\boldsymbol{p}_{1}, {}^{\mathsf{W}}\boldsymbol{p}_{2}, {}^{\mathsf{W}}\boldsymbol{\tilde{v}} \mid {}^{\mathsf{W}}\boldsymbol{\tilde{v}} \in {}^{\mathsf{W}}\boldsymbol{\tilde{V}} \land {}^{\mathsf{W}}\boldsymbol{\tilde{v}} \cdot {}^{\mathsf{W}}\boldsymbol{\hat{z}} > 0 \land {}^{\mathsf{W}}\boldsymbol{\tilde{v}} \cdot {}^{\mathsf{W}}\boldsymbol{\hat{x}} > 0 \}$$

$$(4.39)$$

Wrapping direction & coordinate frame Hence, the half space contains the preceding and succeeding viapoints as well as all mesh vertices that have positive projections on both the *z* and *x* axis of *F*. As the cardinality of *H* affects all subsequent processing steps, false-positive vertices (vertices that are considered to be relevant for the path computation but which are not) should be filtered out. One approach to reduce the cardinality, as described in [23], is to exploit the fact that the path is close to the straight line connecting the preceding and succeeding via-points. Therefore, a limit angle α_{max} can be defined. If the projection of a mesh vertex ${}^W \tilde{v}$ onto the y-z plane of *F* is greater than α_{max} , ${}^W \tilde{v}$ is excluded from *H*. In fact, this reduces the half-space *H* to a triangular prism (see Figure 1 in [23]).

Convex envelope In the following processing step, the convex envelope of the halfspace H is calculated by: (i) computing the convex hull of H (i. e. the smallest convex set that contains all points in H) and (ii) removing the hull facets that are on the inside of the original mesh. Computing the convex hull of a 3D point set is a standard problem in computer graphics and a variety of algorithms exist (e. g. the Quickhull algorithm implemented by the Qhull library³—see also [5]). In the following let us consider that the convex hull C_H of H has been successfully computed and is given by:

$$C_H := \{ ({}^W C_V, C_F, {}^W C_N) \}$$
(4.40)

where ${}^{W}C_{V} := \{{}^{W}c_{V_{1}}, \ldots, {}^{W}c_{V_{o}}\}$ is a set containing the hull points ${}^{W}c_{V} \in \mathbb{R}^{3}, C_{F} := \{c_{F_{1}}, \ldots, c_{F_{p}}\}$ is a set containing the triangulated hull facets $c_{F} \in \mathbb{N}^{3}$ indexing ${}^{W}C_{V}$, and ${}^{W}C_{N} := \{{}^{W}c_{N_{1}}, \ldots, {}^{W}c_{N_{p}}\}$ is a set containing the normal vectors ${}^{W}c_{N} \in \mathbb{R}^{3}$ of the facets. Then the convex envelope C_{E} is given by:

$$C_E := \{ ({}^W C_V, C_F, {}^W C_N) | {}^W \boldsymbol{c}_N \in {}^W C_N \land {}^W \boldsymbol{c}_N \cdot {}^W \boldsymbol{\hat{z}} > 0 \}$$
(4.41)

Hence, the convex envelope is the subset of the convex hull that contains only those facets of which the angle between the facet normal and ${}^{W}\hat{z}$ is in the range $[-90^{\circ}, 90^{\circ}]$.

Geodesic path Once the convex envelope is determined, the shortest muscle path from the preceding to the succeeding via-point can be computed. As both p_1 and p_2 are contained within C_E , this problem is identical to what is known in computational geometry as finding the *geodesic shortest path* on a convex polyhedron (in contrast to the *Euclidean shortest path*, "which may leave the 2-manifold and fly through 3-space" [111]). While computing the Euclidean shortest path is NP-hard [12], many algorithms for finding either the exact geodesic path in polynomial time [13, 65, 66, 88] or approximate solutions with guaranteed error bounds [63, 74] exist (for a survey of existing methods

³ available at http://www.qhull.org

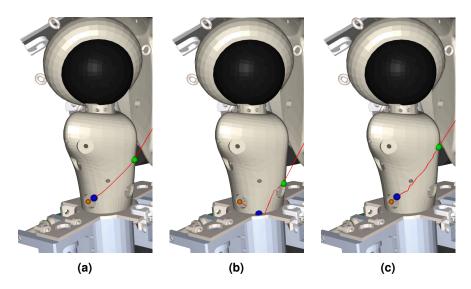


Figure 4.12: Comparison of geodesic path algorithms. Three algorithms are analyzed by comparing the lines of action, the length l_M and the runtime for the path computation *t* for the muscle *Anterior Deltoid* of the physics-based model of the Anthrob robot. (a) Exact geodesic path computed with an optimized implementation of the algorithm described by Mitchel et al. [88] ($l_M = 0.1305 \text{ m}$, $t_{\text{Exact}} = 1.8 \text{ ms}$), (b) An approximation solution using Dijkstra's algorithm on the mesh ($l_M = 0.1347 \text{ m}$, $t_{\text{Dijkstra}} = 0.086 \text{ ms}$), (c) A subdivision approximation which first increases the mesh density by adding N = 10 additional vertices to each edge of the mesh and subsequently runs Dijkstra's algorithm ($l_M = 0.1321 \text{ m}$, $t_{\text{Subdivision}} = 11.3 \text{ ms}$).

also see [78]). However, even though algorithms that compute approximate solutions run typically faster than exact algorithms (e.g. see [148]), the latter must be favored in the current context. The reason is that approximate geodesic algorithms normally only provide good estimates for the path length, but not for the way-points of the path. These, however, are essential for computing the lines of action of the muscle (see below). This problem is illustrated in Figure 4.12 by comparing the exact geodesic path (an optimized implementation of the algorithm described by Mitchell et. al [88]) with two approximations: (i) the shortest path on the mesh using Dijkstra's algorithm [24] and (ii) a subdivision algorithm which first increases the mesh density by adding N additional vertices to each edge of the mesh and then runs Dijkstra's algorithm on the resulting mesh⁴. It can be seen that even though the errors in the total path length of the approximations are relatively small ($\triangle l_M \sim 4 \,\mathrm{mm}$ for the Dijkstra and $\triangle l_M \sim 2 \,\mathrm{mm}$ for the subdivision approximation, respectively), the lines of action of the muscle differ significantly. Particularly, the Dijkstra approximation would induce a totally different joint torque in the spherical glenohumeral joint of the presented model (see Figure 4.12b). The reason for this is that the path is constrained by the mesh vertices and edges and cannot cross facet surfaces. This problem can be partially

⁴ all algorithms were implemented by Daniel Kirsanov—code available from http://code.google.com/p/geodesic/

52 MUSCLE MODELING

compensated for by increasing the mesh density as shown in Figure 4.12c for N = 10 additional vertices per edge. However, the result still deviates from the exact solution and, even more importantly, the runtime for computing the path increased from $t_{\text{Dijkstra}} = 0.086 \text{ ms to}$ $t_{\text{Subdivision}} = 11.3 \text{ ms}$, which is one order of magnitude longer as the exact solution ($t_{\text{Exact}} = 1.8 \text{ ms}$). Therefore, an exact algorithm should be favored to compute the muscle path and, if runtime is an issue, parallelization techniques should be considered.

Muscle lines of action & length

Let *G* be a sequence of vectors *g* describing the geodesic path from p_1 to p_2 :

$$G := \{g_1 = p_1, g_2, \dots, g_n = p_2\} \qquad n \ge 2$$
(4.42)

Then the muscle lines of action for a muscle segment containing a mesh wrapping surface are given by the two tangent points g_2 and g_{n-1} and the resulting four unit vectors (see also Figure 4.11d):

$$\hat{f}_1 = \frac{g_2 - p_1}{\|g_2 - p_1\|}$$
 $\hat{f}_2 = -\hat{f}_1$ (4.43)

$$\hat{f}_3 = \frac{g_{n-1} - p_2}{\|g_{n-1} - p_2\|} \qquad \hat{f}_4 = -\hat{f}_3 \tag{4.44}$$

Furthermore, the length of this muscle segment is given by the Euclidean distance of the *n* path vectors:

$$l_{M'} = \sum_{i=1}^{n-1} \|g_i - g_{i+1}\|$$
(4.45)

4.3 SERIES ELASTIC ELEMENT MODELING

As shown in Figure 4.1, the SEE model relates the expansion l_S to a force magnitude f, which is used to scale the lines of action of the muscle and to set the load of the actuator model. Various types of SEEs can be found in existing musculoskeletal robots, ranging from simple elastic shock cords with stress-strain properties that can be approximated by a linear spring-damper [165] to complex non-linear spring units which employ linear springs and an elegant mechanical routing of the tendon to achieve nonlinearity [82, 112]. But without any loss of generality, only two types of elastic elements, which are relevant for this work, are considered: (i) a linear spring-damper and (ii) SEEs made from polymers, such as the Nitrile Butadiene Rubber (NBR) O-rings used for the Anthrob robot, which exhibit viscoelastic properties reminiscent of the human muscle-tendon unit [32].

4.3.1 Linear Spring-Damper

The most simple SEE model is a linear spring-damper, where a Hookean spring and a Newtonian damper are combined in parallel. Let *k*

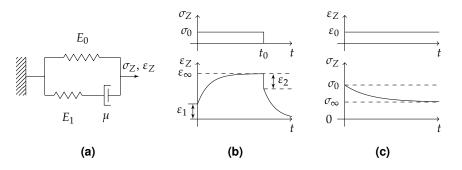


Figure 4.13: Standard linear solid (SLS) or Zener model. (a) The Zener model parallely combines a Hookean spring with a Maxwell fluid model. (b) Creep and creep recovery response. If a stress σ_0 is applied, the Zener model shows an immediate increase in strain equal to $\varepsilon_1 = \sigma_0/(E_0 + E_1)$ and exponentially approaches an equilibrium strain as the dashpot is expanding of $\varepsilon_{\infty} = \sigma_0/E_0$. When the stress is removed, the Zener model shows the inverse behavior—an immediate decrease of strain followed by an exponential decay. (c) Relaxation response. The relaxation response of the Zener model is similar to the Maxwell fluid model (see Figure B.2c) with the stress at time 0 being equal to $\sigma_0 = (E_0 + E_1)\varepsilon_0$ and approaching a non-zero equilibrium of magnitude $\sigma_{\infty} = E_0\varepsilon_0$ after infinite time.

and *c* be the stiffness and damping coefficient of the spring and of the damper, respectively. Furthermore, let l_S be the expansion and dl_S/dt be the velocity of the spring-damper, then the force exerted by the linear spring damper is equal to:

$$f = k l_S + c \frac{dl_S}{dt} \quad \text{with } f \ge 0 \tag{4.46}$$

Note that muscles can only pull and not push. Hence, the additional constraint of $f \ge 0$.

4.3.2 Models of Linear Viscoelasticity

In this section, two models of linear viscoelasticity are presented: (i) the standard linear solid (or Zener) model and (ii) the Wiechert model. Both models rely on the serial and parallel assembly of Hookean springs and Newtonian dampers and are relevant for the modeling of the NBR O-rings used in the Anthrob robot as SEE. The reader not familiar with these basic mechanical elements, with their serial and parallel assembly known as the Maxwell and Voigt model, or with mechanical stress and strain in general is referred to Appendix B for a brief introduction.

While the Maxwell fluid model presented in Appendix B mimics the relaxation properties of polymers, it fails to capture the creep properties found in polymer experiments. On the other hand, the Voigt model successfully emulates the creep response found in polymers but fails to mimic the relaxation behavior. The standard linear solid (SLS) or Zener model solves these problems by leveraging either the

Standard linear solid (or Zener) model

54 MUSCLE MODELING

Voigt solid model with a serial spring—the Voigt form of the SLS model [4, 10]—or by extending the Maxwell fluid model with a parallel, linear spring as shown in Figure 4.13a. This variant of the SLS model, which is typically referred to as the Maxwell form of the SLS [4, 18] or Zener model [131], should be preferred over the Voigt form of the SLS if the stress is to be computed from a strain input. The differential equation of the Zener model is given by [18]:

$$\dot{\sigma}_Z + \frac{E_1}{\mu} \sigma_Z = \dot{\varepsilon}_Z (E_0 + E_1) + \frac{E_0 E_1}{\mu} \varepsilon_Z$$
 (4.47)

where σ_Z is the stress, ε_Z the strain, μ the viscosity of the dashpot and E_0 as well as E_1 are the Young's moduli of the two springs. The response of the Zener model to creep and relaxation tests is shown in Figure 4.13b and 4.13c, respectively. In the creep test, the Zener model shows an immediate increase in strain ε_1 in response to the applied stress σ_0 , equal to:

$$\varepsilon_1 = \frac{\sigma_0}{E_0 + E_1} \tag{4.48}$$

This step-response is followed by an exponential increase in strain due to the expanding dashpot, similar to the Voigt model (see Figure B.3b). As the strain of the spring in the Maxwell branch decreases when the dashpot strain is increasing, the equilibrium strain of the Zener model ε_{∞} is equal to:

$$\varepsilon_{\infty} = \frac{\sigma_0}{E_0} \tag{4.49}$$

Finally, the creep recovery response of the Zener model shows the inverse behavior of the creep case. Upon stress removal, the strain immediately decreases by ε_2 . However, the magnitude of ε_2 depends on the current expansion of the spring in the Maxwell branch at the time when the stress is removed and is bound by:

$$0 \le \varepsilon_2 \le \frac{\sigma_0}{E_0 + E_1} \tag{4.50}$$

In the relaxation test, the Zener model exhibits a similar response as the Maxwell fluid model (see Figure B.2c) with two differences. First, the initial stress σ_0 at time 0 is equal to the sum of the stresses of the two springs:

$$\sigma_0 = (E_0 + E_1) \varepsilon_0 \tag{4.51}$$

And second, the equilibrium stress σ_{∞} is non-zero and given by:

$$\sigma_{\infty} = E_0 \,\varepsilon_0 \tag{4.52}$$

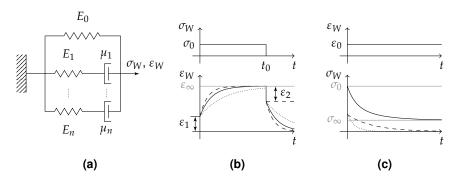


Figure 4.14: Wiechert model. (a) The Wiechert model combines a linear spring with multiple Maxwell fluid models in parallel to achieve a better fitting of the timedependent properties of viscoelastic solids. (b) Creep and creep recovery response. (c) Relaxation response. The dashed and dotted lines in panels (b) and (c) show the contribution of two Maxwell terms to the stress and strain trajectories.

Let us consider again the first order differential equation of the Zener model given by Equation 4.47. From a numerical point of view, the strain derivative contained within the equation can be problematic and induce simulation instabilities. But, as we will see, this derivative can be eliminated by a change of variables [144]. Let σ_Z and $\dot{\sigma}_Z$ be equal to:

$$\sigma_Z = (E_0 + E_1)\,\varepsilon_Z - y \tag{4.53}$$

$$\dot{\sigma}_Z = (E_0 + E_1) \dot{\varepsilon}_Z - \dot{y}$$
 (4.54)

Then substituting Equation 4.53 and Equation 4.54 into Equation 4.47, yields:

$$E_{\text{tot}} \dot{\varepsilon}_{Z} + \frac{E_{0}E_{1}}{\mu} \varepsilon_{Z} = E_{\text{tot}} \dot{\varepsilon}_{Z} - \dot{y} + \frac{E_{1}}{\mu} (E_{\text{tot}} \varepsilon_{Z} - y)$$
with $E_{\text{tot}} = E_{0} + E_{1}$

$$(4.55)$$

Rearranging finally yields:

$$\dot{y} + \frac{E_1}{\mu} y = \frac{E_1^2}{\mu} \varepsilon_Z$$
 (4.56)

Hence Equation 4.47 was transformed into a first order linear differential equation which is numerically more stable than Equation 4.47 as it only contains the strain as dependent variable.

Wiechert model

The experimentally observed creep and relaxation behaviors of polymers are normally not adequately fit if only a single relaxation and retardation time is used as in the Zener model. Hence, the Wiechert model combines a generalized Maxwell model-an arbitrary number of parallel Maxwell fluid models [10, 139]—and a Hookean spring as shown in Figure 4.14a to provide a better fit of the experimentally found creep and relaxation times. In the Wiechert model, the total

Numerical stability of the Zener model

56 MUSCLE MODELING

stress σ_W is equal to the sum of the stress of the spring σ_s and the *n* Maxwell terms:

$$\sigma_W = \sigma_s + \sum_{i=1}^n \sigma_{M_i} \tag{4.57}$$

whereas the strain ε_W is identical for all branches due to the parallel design of the model:

$$\varepsilon_W = \varepsilon_s = \varepsilon_{M_1} = \ldots = \varepsilon_{M_n}$$
(4.58)

Substituting the basic constitutive equations for the spring and the Maxwell fluid model (Equation B.6 and B.10, respectively) into Equation 4.57 and 4.58 and solving for σ_W yields an inhomogeneous linear differential equation of order *n*. Under the assumption that at t = 0 a constant strain ε_0 is applied—as in the relaxation test—the solution of this differential equation is equal to:

$$\sigma_{W} = \left[\sum_{i=1}^{n} E_{i} \exp\left(-t/\tau_{i}\right) + E_{0}\right] \varepsilon_{0}$$
(4.59)

which is simply the sum of the stress induced by the free spring and of the *n* Maxwell terms (the term τ_i is the relaxation time constant of the *i*th Maxwell term—see Appendix B). It should be noted, that solving this differential equation for the strain, as required for the creep test, is more complicated and neglected in the context of this work. In that case, the Voigt form of the Wiechert model—a Hookean spring in series with *n* Voigt models—should be preferred over the Wiechert model [10]. Qualitative responses of the Wiechert model with two Maxwell terms to creep, creep recovery and relaxation experiments are shown in Figure 4.14b and 4.14c, respectively.

4.4 MUSCLE FRICTION

In the muscle kinematics models presented so far, friction was neglected which allowed for the computation of the muscle path by computing the shortest path based on geometrical considerations. However, frictional muscle forces play an important role in the dynamics of musculoskeletal robots [58] and two types of muscle friction forces can be differentiated: (i) forces that alter the muscle path and (ii) forces that do not. While for the latter the muscle path is still equivalent to the shortest path, this is not the case for the former. Hence, the friction forces which alter the path cannot be modeled by geometric considerations alone, but require a physical representation of the muscle within the dynamics engine (e.g. using soft-body dynamics). Preliminary tests of such soft-body muscles have revealed two main problems [165]: (i) a high computational load, impeding real-time simulations of even simple models, and (ii) simulation instabilities due

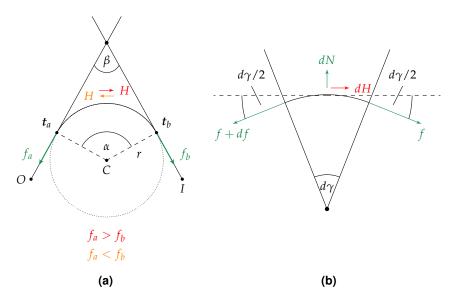


Figure 4.15: Coulomb friction on via-points and spherical wrapping surfaces. Coulomb friction on via-points and spherical wrapping surfaces can be modeled by the capstan equation. (a) Geometric relations of friction forces on spherical obstacles. (b) Infinitesimal segment of the arc shown in panel (a).

to the collisions between the muscle and the skeleton—particularly in more complex setups. Therefore, soft-body muscles were not used in this work and only friction models, that do not alter the muscle path, are considered in this section.

Let us consider the setup shown in Figure 4.15a comprising the muscle origin *O*, its insertion point *I* and a spherical obstacle. Then, in accordance with Coulomb's law of friction, two types of friction can be observed for the sphere surface [40, 120]: (i) static friction (or stiction), which prevents the muscle from sliding over the sphere, and (ii) kinetic friction, which is the resisting force that acts as soon as the static friction force has been overcome by the net external force and the muscle starts sliding. These friction forces can be modeled by what is known in the literature as Euler-Eytelwein's formula [40] or capstans equation [147].

Let us assume that the forces f_a and f_b in Figure 4.15a are given *Capstal* and that $f_a > f_b$. Furthermore, let us consider an infinitesimal small segment of the circle with angle $d\gamma$ as shown in Figure 4.15b. Then, by assuming balance of forces, the following two relations can be derived for the friction force dH and normal force dN [40]:

$$dH = (f + df)\cos\frac{d\gamma}{2} - f\cos\frac{d\gamma}{2}$$
(4.60)

$$dN = f \sin \frac{d\gamma}{2} + (f + df) \sin \frac{d\gamma}{2}$$
(4.61)

Coulomb friction on obstacles

Capstan equation

As $d\gamma$ is infinitesimal, $\cos(d\gamma/2) \approx 1$, $\sin(d\gamma/2) \approx d\gamma/2$ and the product $df d\gamma$ can be neglected. This simplifies Equation 4.60 and Equation 4.61 to:

$$dH = df \tag{4.62}$$

$$dN = f d\gamma \tag{4.63}$$

Now, in accordance with Coulomb's law of friction the friction force *H* is proportional to the normal force *N* with the proportionality constant given by the friction coefficient μ :

$$H = \mu N \tag{4.64}$$

Hence, by applying Equation 4.64 to Equation 4.62 and Equation 4.63, we can eliminate the normal force:

$$df = \mu f d\gamma \tag{4.65}$$

Rearranging yields:

$$\frac{df}{f} = \mu d\gamma \tag{4.66}$$

Finally, integrating over the complete wrapping angle α provides the capstan equation which relates the forces on both sides of the spherical obstacle:

$$\int_{f_b}^{f_a} \frac{1}{f} df = \int_0^\alpha \mu d\gamma \tag{4.67}$$

$$\ln \frac{f_a}{f_b} = \mu \alpha \tag{4.68}$$

$$f_a = f_b \, e^{\,\mu\alpha} \tag{4.69}$$

Now let us first consider the static friction case. In accordance with Coulomb's law of friction, the static friction force H_s is equal to:

$$H_s \le \mu_s N \quad \text{with } H_s \in [0, \mu_s N]$$

$$(4.70)$$

where μ_s is the static friction coefficient. Then the muscle is not sliding on the surface (v = 0) as long as the following inequalities are satisfied:

$$f_a \le f_b \, e^{\,\mu_s \alpha} \quad \text{if } f_a > f_b \ \land \ v = 0 \tag{4.71}$$

$$f_a \ge f_b e^{-\mu_s \alpha} \quad \text{if } f_a < f_b \ \land \ v = 0 \tag{4.72}$$

As mentioned previously, static friction transforms into kinetic friction H_k once the static friction is overcome and the muscle starts sliding on the sphere with $v \neq 0$. But, as H_k is opposing the net external

Static friction on spherical obstacles

Kinetic friction on spherical obstacles

forces, we have to consider again two cases: (i) the muscle is sliding on the surface in the direction of the origin (v < 0) and (ii) the muscle is sliding in the direction of the insertion (v > 0). These considerations yield the following two equations for relating the muscle forces:

$$f_a = f_b e^{\mu_k \alpha} \quad \text{if } v < 0 \tag{4.73}$$

$$f_a = f_b e^{-\mu_k \alpha}$$
 if $v > 0$ (4.74)

where μ_k is the kinetic friction coefficient.

When numerically simulating static and kinetic friction, three problems are encountered: (i) H_s is discontinuous at v = 0 as it can take any value in the range $[0, \mu_s N]$, (ii) the fact that μ_s is typically higher than μ_k [40, 85] introduces a second discontinuity at the transition from static to kinetic friction and (iii) the friction force changes its sign as it always opposes the net external force. These problems can be solved by using a continuous function for both the sign of the force and the transition between static and kinetic friction and by approximating the static friction force in the zero velocity vicinity via a linear or sigmoid function [1]. Thus, if the previously introduced Stribeck friction curve [146] is used to model the static-to-kinetic friction transition and the logistic function for the zero velocity approximation, the resulting continuous friction coefficient μ_{s+k} is given by:

$$\mu_{s+k} = \left[\frac{2}{1 + e^{-\kappa v/v_s}} - 1\right] \left[\mu_k + (\mu_s - \mu_k)e^{-|v|/v_s}\right]$$
(4.75)

where v_s is the Stribeck velocity and κ a parameter defining the steepness of the approximation around v = 0. This results in the following final equation for the friction on spherical obstacles:

$$f_a = f_b \, e^{\,\mu_{s+k}\,\alpha} \tag{4.76}$$

Hence, static and kinetic friction as well as the two cases considered for each of the friction types previously were combined in a single, continuous equation.

If spherical wrapping surfaces are considered, Equation 4.76 can be applied directly by replacing α with the wrapping angle α_{Λ} (see Equation 4.26). However, if via-points are considered, Equation 4.76 still holds but α has to be computed from the angle β between the preceding and succeeding muscle segments (see also Figure 4.15a):

$$\alpha = \pi - \beta \tag{4.77}$$

Combined static and kinetic friction simulation model

Friction on spherical wrapping surfaces and via-points

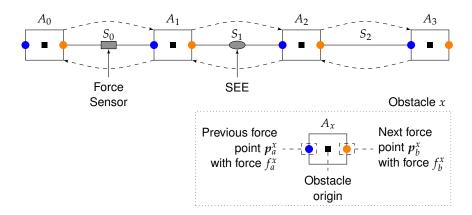


Figure 4.16: Proposed data structure of a muscle in a physics-engine. In the proposed data structure, a muscle is represented by a doubly linked list of *n* muscle obstacles *A*, which result in n - 1 muscle segments *S*. Furthermore, each obstacle *A* is defined by a 6-tuple of the form $A := ({}^{B}T_{A}, B, p_{a}, p_{b}, f_{a}, f_{b})$, where ${}^{B}T_{A}$ is a transformation matrix defining the position and orientation of the obstacle in the frame of body *B*, $p_{a/b}$ are the previous and next force points and $f_{a/b}$ the force magnitudes of the previous and next muscle segments.

4.5 MUSCLE DYNAMICS ALGORITHM

In physics-based simulation engines, the dynamics of a robot are simulated by (i) applying forces to the rigid bodies of the robot and (ii) iteratively solving constraint equations to simulate the resulting joint motions (see Chapter 3). Therefore, physics-engines can be considered to operate in the *body space*. This differs significantly from classical forward dynamic simulation approaches in robotics where joint accelerations are computed directly from joint torques (see Section 2.2). However, as we will see in this section, the body-space approach is particularly advantageous in the case of musculoskeletal robots as it allows us to derive a universal algorithm for simulating the dynamics of musculo-skeletal interactions which is independent of the type of wrapping surfaces within the muscle path, the used SEE and the type of actuator.

Every type of wrapping surface can be represented by two force points In Section 4.2 muscle kinematics models of varying complexity have been presented. The analysis of the forces applied by each model reveals that every type of wrapping surface can be represented by two straight-line muscle segments: one from the preceding via-point to the point where the muscle *touches* the surface and a second one from the point where the muscle *leaves* the surface to the succeeding via-point. Hence, by computing these *force points* as well as the force magnitudes of the muscle segments and applying the resulting forces to the rigid bodies of the force points, we obtain an algorithm for simulating the musculo-skeletal interactions of muscles which is independent of the defined via-points and/or wrapping surfaces.

Algorithm 4.1: Algorithm for the physics	-based simulation of muscle dynamics.
Data: n obstacle points A (via-points, s	spherical wrapping surfaces, etc.),
segment index j of the force sen	sor, segment index k of SEE, time since
the last simulation step $ riangle t$	
for $i \leftarrow 0$ to $n-1$ do	<pre>// update obstacle origins</pre>
${}_{2} {}^{W}T_{A}^{i} = {}^{W}T_{B}^{i} {}^{B}T_{A}^{i};$	
3 end	
4 for $i \leftarrow 0$ to $n-1$ do	// update force points $oldsymbol{p}_a^i$ and $oldsymbol{p}_b^i$
5 updateForcePoints();	
6 end	
$_7 l_M \leftarrow 0.0;$	
s for $i \leftarrow 1$ to $n-1$ do	<pre>// compute muscle length</pre>
9 $ l_M \leftarrow l_M + \texttt{getLength}(p_b^{i-1}, p_b^i);$	
10 end	
11 $f_b^k, f_a^{k+1} \leftarrow getSEEForce(l_M - l_T, \triangle t);$	<pre>// compute SEE force</pre>
12 for $i \leftarrow k$ to 0 do	<pre>// propagate force to muscle origin</pre>
$f_a^i \leftarrow applyFriction(f_b^i);$	
applyForce(B^i, p_a^i, f_a^i);	
applyForce $(B^i, p_b^i, f_b^i);$	
16 end	
17 for $i \leftarrow k+1$ to $n-1$ do //	propagate force to muscle insertion
18 $f_h^i \leftarrow applyFriction(f_a^i);$	
applyForce $(B^i, p_a^i, f_a^i);$	
applyForce $(B^i, p_h^i, f_h^i);$	
21 end	
22 $f_S \leftarrow f_h^j$;	// assign force sensor force f_{S}
$l_{23} l_T \leftarrow \text{integrateActuator}(u, f_b^0, \bigtriangleup t);$	// update tendon length l_T

The proposed data structure for the muscle dynamics algorithm is shown in Figure 4.16. The muscle path from the muscle origin to its insertion is represented by a doubly linked list of n muscle obstacles A (via-points or wrapping surfaces) which result in n - 1 muscle segments S as well as by the two scalars j and k which define the segment index of the force sensor and of the SEE, respectively. Furthermore, each muscle obstacle is defined by a 6-tuple of the form:

$$A := ({}^{\scriptscriptstyle B}\boldsymbol{T}_A, \boldsymbol{B}, \boldsymbol{p}_a, \boldsymbol{p}_b, f_a, f_b) \tag{4.78}$$

where ${}^{B}T_{A}$ is the transformation matrix defining the position and orientation of the obstacle in the frame of reference of the body *B*, p_{a} and p_{b} are the preceding and succeeding force points and $f_{a/b}$ are the force magnitudes of the corresponding muscle segments.

The algorithm for simulating the muscle dynamics is given in Algorithm 4.1. It proceeds as follows:

- 1. The world positions and orientations of all obstacle origins are updated (lines 1-3).
- 2. The preceding and succeeding force points *p*_{*a*} and *p*_{*b*} are computed (lines 4-6).

Algorithm for simulating muscle dynamics

Muscle data structure

- 3. The length of the muscle l_M is calculated (lines 7-10).
- 4. The expansion of the SEE and hence the force *f* is computed and set for the muscle segment *k* of the SEE (line 11).
- 5. f is applied to the rigid bodies of the obstacles. Possible friction forces are considered by successively propagating f from k to the muscle origin and insertion (lines 12-19).
- 6. The force sensor force is assigned by taking the force sensor segment index *j* into account (line 20).
- 7. Finally, the tendon length l_T is updated by integrating the actuator model (line 21).

The functions in Algorithm 4.1 (e.g. updateForcePoints) are pure virtual functions which have to be implemented by the concrete type of obstacle (e.g. spherical wrapping surface), SEE and actuator.

4.6 **DISCUSSION**

Role of tendon radius in muscle kinematics In this chapter, the modeling of muscle kinematics within the physicsbased framework was introduced. But so far, the tendon radius was not taken into account by the presented models. This simplification is reasonable when the tendon radius is small but results in nonnegligible errors for thick tendons due to the offset in the force points. In these cases, a readjustment of the force points to match the tendon center is required. This can be achieved either online or offline by manually adjusting the via-point origins and by inflating the wrapping surfaces. However, the tendon radius not only affects the muscle kinematics but also the actuator dynamics model. Here, the tendon radius increases the spindle radius and therefore the linear velocity of the tendon. Hence, when this is a concern, the tendon radius should be added to the spindle radius.

Stiff muscles In some musculoskeletal robots stiff muscles, i.e. muscles without SEE are used. These muscles cannot be modeled accurately in the physics-based framework as the force of the SEE is a prerequisite for the computation of the actuator load as well as for the forces applied to the rigid bodies of the skeleton. Hence, if a perfectly stiff muscle is considered, two questions arise: (i) what is the force applied by the muscle and (ii) what is the magnitude of the actuator load? While for the former the output force of the actuator can be used, the latter cannot be determined as it depends on a variety of factors, such as the number and the mass of the rigid bodies affected by the muscle. Hence, stiff muscles can only be approximated. Typically, an SEE with a high spring stiffness is an adequate substitute.

In anatomy, a ligament denotes passive fibers attached to two neighboring bones that constrain the joint motion [134]. Hence, in the presented framework, a ligament can be simulated as a passive muscle, i.e. a muscle without actuator and with a constant tendon length.

Muscle friction plays an important role in the muscle dynamics as shown in [58]. Hence, a friction model based on the capstan equation was introduced. However, this friction model only alters the force magnitude but not the muscle path. The reason is that all presented muscle kinematics algorithms rely on geometrical considerations of the path and do not take physical properties of the muscle, such as the mass of the tendon, into account. While, at first glance, this seems to be a serious limitation of the chosen approach, the simulation-reality gap measured for the physics-based model of the Anthrob proves the opposite (see Chapter 8). Ligaments

Muscle friction

5

MODEL CALIBRATION

Today, computer simulations are available for many different usecases in science and engineering. But independent of the use-case and the model complexity, every simulation exhibits a more or less prominent simulation-reality gap. There are two main sources for these discrepancies: (i) model errors and (ii) parameter errors. Model errors result from the formulation of the used models and are difficult to fix. If, for instance, the muscle kinematics are modeled by a straightline connection and muscle wrapping occurs in the robot, the model will never be able to faithfully reproduce the outputs of the real system (e.g. muscle force or joint position trajectories). Parameter errors, in contrast, are errors that result from the wrong parametrization of the models. These errors can be minimized by adjusting the model parameters to match the input-output behavior of the real system. This process, which is illustrated in Figure 5.1, is called model calibration [69]. In this chapter, a model calibration method for the optimization of the statics (i.e. the simulation-reality gap of equilibrium postures) and dynamics (i.e. the simulation-reality gap of trajectories) of physics-based, musculoskeletal robot models is presented. The calibration method uses an ES to minimize the joint position and muscle force error of the model. These output variables have been selected as they are particularly important for deploying the physics-based model as an offline virtual test-bed for joint or operational space controllers or as an online internal model for robot control.

The used ES algorithm is presented in Section 5.1. As we will see, evolution strategies are particularly suited for this use-case as the optimization algorithm can be applied directly to the physics-based model and no analytical model representation is required. Furthermore, the ES procedure is generic in the sense that the algorithm is independent of the used input and output variables. This is achieved by using a problem-specific objective function that evaluates inputs to the model and computes the model error. Two objective functions have been developed to calibrate: (i) the statics and (ii) the dynamics of musculoskeletal robots. These objective functions are presented in Section 5.2 and Section 5.3, respectively. The presented calibration method has been evaluated by applying it to the physics-based model of the Anthrob robot in Chapter 8.

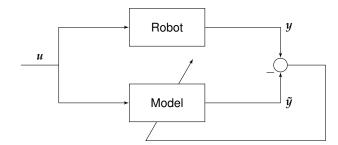


Figure 5.1: Principle of parametric model calibration. The simulation-reality gap of a model can be minimized by means of model calibration. Therefore, robot and model outputs (y and \tilde{y} , respectively) are compared for inputs u and the model parameters are adjusted to reduce the error.

5.1 EVOLUTION STRATEGY

An ES is proposed to calibrate the physics-based models of musculoskeletal robots. Evolution strategies, which have emerged in the late 1960s and early 1970s in Germany [125, 137], take inspiration from Biology by replicating the evolutionary processes "underlying birth and death, variation and selection, in an iterative, respectively generational, loop" [6]. Evolution strategies belong to the larger class of Evolutionary Algorithms (EA) and can be applied to real-valued as well as discrete search spaces. A variety of evolution strategies has evolved in the past 40 years and, even though some best practices exist, the selection of the proper methods and particularly of the model parameters involved in the optimization highly depends on the problem at hand. Hence, in the remainder of this section, the proposed ES algorithm is presented. For a comprehensive review on evolution strategies the interested reader is referred to [6] or [163].

The Evolution Strategy (ES) algorithm As ES, a *comma* selection variant of the form (μ, λ) is chosen, where μ and λ are the population sizes of the parent and offspring populations, respectively. In contrast to *plus* selection $(\mu + \lambda)$, the *comma* selection uses only the offspring population as selection pool and has been shown to be preferred over the *plus* variant for unbounded, real-valued search spaces [136]. The parent and offspring populations (\mathcal{P}_p and \mathcal{P}_o , respectively) consist of individuals \mathcal{I} of the form:

$$\mathcal{I}_k := (\boldsymbol{y}_k, \boldsymbol{s}_k, \mathcal{F}_k, \boldsymbol{v}_k) \tag{5.1}$$

where $y_k \in \mathbb{R}^n$ are the *object parameters* modified by the mutation, $s_k \in \mathbb{R}^m$ are the endogenous *strategy parameters* defining the step-size of the mutation (in contrast to the exogenous strategy parameters μ and λ , which are constant for a run), $\mathcal{F}_k \in \mathbb{R}$ is the objective function value or *fitness* and v_k is a Boolean flag, which is true if individual kis *valid* and false otherwise (see *Evaluation* below). As mentioned in the introduction of this section, optimization by means of evolution strategies is performed by a cyclic execution of a series of operations.

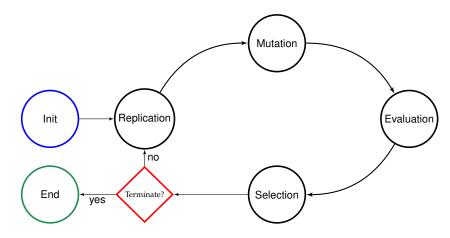


Figure 5.2: Evolution Strategy. Evolution strategies optimize an objective function value by stochastic adaptation of a set of object parameters. The optimization proceeds in an iterative, generational loop inspired by the adaptive processes known from evolution.

One such cycle, called a *generation*, is shown in Figure 5.2 for the proposed ES. It consists of four main operations, each described in detail in the following paragraphs: (i) replication, (ii) mutation, (iii) evaluation and (iv) selection. Note that no recombination operator is used. As *comma* selection strategies only use the offspring population as selection pool, the best individual found during the generational loop does not survive. Hence, in addition to \mathcal{P}_p and \mathcal{P}_o , the individual $\mathcal{I}_{\text{best}}$ is introduced, which is used to store the best individual of all generations and which is returned when the algorithm terminates. For a pseudo-code notation of the algorithm, the interested reader is referred to Algorithm 5.1.

Before the ES generational loop can be started, the algorithm must be initialized. Hence, the generation counter g is set to zero and the parent population of the first generation \mathcal{P}_p^{0} is created. Therefore, an *ancestor* individual $\mathcal{I}_A := (y_A, s_A, \mathcal{F}_A, v_A)$, is created where each $y_i \in y_A$ represents a real-valued quantity of the physics-based model (e. g. the x-coordinate of a via-point) and each $s_i \in s_A$ the initial, userdefined step-size of the mutation operator (see below). Furthermore, v_A is set to true and \mathcal{F}_A to infinity. This ancestor individual is subsequently replicated μ times to create \mathcal{P}_p^{0} .

The μ individuals of the parent population \mathcal{P}_p^{g} are replicated to form the offspring population \mathcal{P}_o^{g} . Therefore, the parent population is first sorted in ascending order based on \mathcal{F} . Then, the offspring population is created by iterating over the μ individuals of the sorted list in ascending order and cloning the individuals until λ individuals are created.

Initialization

Replication

68 MODEL CALIBRATION

Mutation The mutation operation defines the way the search space is explored and hence has a huge impact on the convergence rate of the calibration. As mutation operator, a Gaussian-based, non-isotropic mutation operator (one dedicated mutation step-size per object parameter) with strategy parameter self-adaptation is used [6]. The selfadaptation heuristic of individual *k* is given by:

$$\tilde{s}_{k}^{\ g} = s_{k}^{\ g} \exp\left(\frac{\mathcal{N}^{k}(0,1)}{\sqrt{2n}}\right) \begin{bmatrix} \exp\left(\frac{\mathcal{N}^{s}(0,1)}{\sqrt{2\sqrt{n}}}\right) \\ \vdots \\ \exp\left(\frac{\mathcal{N}^{s}_{m}(0,1)}{\sqrt{2\sqrt{n}}}\right) \end{bmatrix}$$
(5.2)

where $\mathcal{N}(0,1)$ is a random value generated by a Gaussian distribution with mean 0 and standard deviation 1. \mathcal{N}^s is re-generated for each strategy parameter of individual k, whereas \mathcal{N}^k is picked only once per generation and individual. The mutated strategy parameters $\tilde{s}_k^{\ g}$ are subsequently used for the mutation of the object parameters in accordance to:

$$\tilde{\boldsymbol{y}}_{k}^{g} = \boldsymbol{y}_{k}^{g} + \begin{pmatrix} \mathcal{N}(0, \tilde{s}_{1}^{g}) \\ \vdots \\ \mathcal{N}(0, \tilde{s}_{n}^{g}) \end{pmatrix}$$
(5.3)

To ensure that the mutated object parameter values do not exceed a specified, reasonable value range (e.g. friction coefficients have to be positive), value range limits are defined for each object parameter and the mutation operation is repeated until a valid object parameter value is obtained.

- *Evaluation* Now the fitness \mathcal{F}_i of each offspring individual $i \in \mathcal{P}_o^g$ is computed by evaluation of the object parameters y_i (the details depend on the used objective function—see Section 5.2 and Section 5.3). During the evaluation it is possible that an error occurs, such as a via-point penetrating a cylindrical wrapping surface. In these cases, the individual is marked as *invalid* by setting $v_k = false$. Invalid individuals are not considered in the subsequent selection operation.
- Selection & The parent population of the next generation \mathcal{P}_p^{g+1} is determined by selecting the μ best (lowest objective function value) and valid individuals from the offspring population \mathcal{P}_o^{g} . If no valid individuals exist or if either one of the following resource and convergence criteria is satisfied, the algorithm terminates:

g

$$\geq g_{\max}$$
 (5.4)

$$\mathcal{F}(\mathcal{I}_{\text{best}}) < \mathcal{F}_{\text{target}} \tag{5.5}$$

where g_{max} and $\mathcal{F}_{\text{target}}$ are the user-defined maximum number of generations and target fitness, respectively. Otherwise, the algorithm continues with the replication operation.

Algorithm 5.1: Evolution strategy algorithm. First the generation counter *g*, the ancestor individual \mathcal{I}_A , the offspring population $\mathcal{P}_p^{\ 0}$ as well as the individual \mathcal{I}_{best} , storing the best individual of all generations, are initialized (lines 1-6). Then the strategy as well as the object parameters of each individual are mutated (lines 9-10). These mutated individuals are then evaluated in the physics-based simulation to compute the fitness \mathcal{F}_i (line 11). Lines 12-14 ensure that the best individual of all generations can be returned when the algorithm terminates. Finally, the new offspring generation is created by selection & replication of the μ best and valid individuals of $\mathcal{P}_o^{\ g}$ (lines 16-18) and the generation counter is incremented (line 19).

Data: a set y_A of object parameters, a set s_A of strategy parameters, the maximum number of generations g_{max} , the target fitness \mathcal{F}_{target}

```
1 g \leftarrow 0;
  <sup>2</sup> \mathcal{I}_{\mathrm{A}} \leftarrow (y_A, s_A, \infty, \texttt{true});
  _{3} foreach \mathcal{I} \in \mathcal{P}_{o}^{0} do
       \mathcal{I} \leftarrow \mathcal{I}_{\mathrm{A}};
  5 end
  6 \mathcal{I}_{\text{best}} \leftarrow \mathcal{I}_A;
      while g < g_{max} \land \mathcal{F}(\mathcal{I}_{best}) > \mathcal{F}_{target} do
 7
                 foreach i \in \{1, \ldots, \lambda\} do
                            \tilde{s}_i^{\ g} \leftarrow \mathsf{mutate}(s_i^{\ g});
  9
                           \tilde{\pmb{y}}_i^{\ g} \leftarrow \texttt{mutate}(\pmb{y}_i^{\ g}, \tilde{\pmb{s}}_i^{\ g});
10
                           \mathcal{F}_i \leftarrow \text{evaluate}(\tilde{y}_i^g);
11
                           if \mathcal{F}_i < \mathcal{F}(\mathcal{I}_{best}) then
12
                                      \mathcal{I}_{\text{best}} \leftarrow \mathcal{I}_i;
13
                           end
14
                 end
15
                 sort(P_o^g);
16
                \mathcal{P}_p^{g+1} \leftarrow \texttt{select}(\mathcal{P}_o^g, \mu);
17
                 \mathcal{P}_o^{g+1} \leftarrow \mathsf{replicate}(\mathcal{P}_p^{g+1}, \lambda);
18
                g \leftarrow g + 1;
19
20 end
```

5.2 STATICS CALIBRATION

In the previous section, a generic ES for calibrating the physics-based models of musculoskeletal robots was presented. However, the algorithm did not define the objective function $\mathcal{F}(\mathcal{I})$ required to compute the fitness \mathcal{F} of an individual \mathcal{I} , neither did it discuss the data used as input for the calibration. Hence, in this section a procedure for the calibration of the model statics will be described by: (i) introducing an objective function that is capable of *guiding* the exploration of the search space and (ii) by discussing the input data required for the calibration of the statics.

Let us consider a single revolute joint actuated by *m* compliant muscles with constant tendon lengths $l_T \in \mathbb{R}^m$. Once all dynamic effects have settled, these constant tendon lengths eventually lead to equilibrium forces $f_{\infty} \in \mathbb{R}^m$ and equilibrium lever arms $l_{A_{\infty}} \in \mathbb{R}^m$. Fi-

Equilibrium points & statics calibration

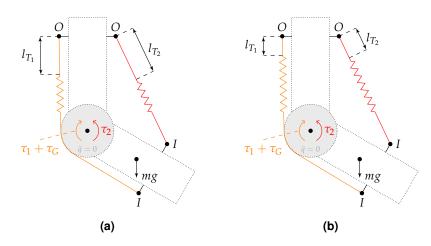


Figure 5.3: Tendon length to joint position ambiguity example. Due to the antagonistic, compliant muscle configuration of musculoskeletal robots, different tendon lengths can lead to identical joint positions. (a) Long tendon configuration. (b) Short tendon configuration. In both configurations the muscle torques and the gravitational torque cancel each other out ($\ddot{q} = 0$) which eventually yields the same joint position.

nally, these equilibrium forces and lever arms result in an equilibrium torque τ_{∞} for the joint, which is equal to the gravitational torque τ_G :

$$\tau_{\infty} = \tau_G = l_{A_{\infty}} \cdot f_{\infty} \tag{5.6}$$

This property of musculoskeletal robots, which is reminiscent of the equilibrium-point control hypothesis from Neurosciences [38], can be exploited to calibrate the statics of the robot.

Objective function Let us consider a robot comprising only revolute joints¹ and let us further consider a set Q of k equilibria E:

$$Q := \{E_1, \dots, E_k\} \tag{5.7}$$

Moreover, let each E_i be given by a triple of the form:

$$E_i := (l_{T_i}, f_i, q_i) \quad i \in \{1, \dots, k\}$$
(5.8)

where $l_{T_i} \in \mathbb{R}^m$ is a vector of tendon lengths, $f_i \in \mathbb{R}^m$ a vector of muscle forces and $q_i \in \mathbb{R}^n$ a vector of joint positions. Then the simulationreality gap of the physics-based model statics can be quantified by: (i) recording a set of equilibria Q^{rob} for the robot, (ii) create the corresponding set Q^{sim} for the simulation by setting the tendon lengths of each E_i^{rob} in the simulation, wait for the dynamics to settle and record E_i^{sim} . Finally compute the joint position and muscle force errors for each pair of robot and simulator equilibria $(E_i^{\text{rob}}, E_i^{\text{sim}})$. Hence, the objective function $\mathcal{F}_S(\mathcal{I})$, for calibrating the model statics, is given by:

$$\mathcal{F}_{S}(\mathcal{I}) = \left[\sum_{i=1}^{k} \kappa_{q} \, \frac{\|\boldsymbol{q}_{i}^{\text{rob}} - \boldsymbol{q}_{i}^{\text{sim}}\|}{n} + \kappa_{f} \, \frac{\|\boldsymbol{f}_{i}^{\text{rob}} - \boldsymbol{f}_{i}^{\text{sim}}\|}{m}\right] / k \tag{5.9}$$

1 other joint types are discussed below

where κ_q and κ_f are coefficients that can be used to adjust the contribution of the joint position and muscle force errors to the fitness of the individual.

The question might arise why the forces of the muscles are considered for the objective function and not just the joint positions. The reason is, that the mapping between tendon lengths l_T and joint positions q is not injective, as different l_T can lead to identical joint positions. This is illustrated in Figure 5.3 where a revolute joint, actuated by two compliant muscles, is shown for two tendon length configurations. While the tendon lengths differ, the resulting joint position would be identical. In fact, due to the compliance of the muscles, there is an infinite number of tendon length vectors that would result in the identical joint position. Hence, neglecting the muscle forces during calibration would eventually lead to low joint position errors but to large muscle force errors. This, however, is not desirable if the model is supposed to be used as a test-bed for robot control algorithms that rely on muscle forces or if the model is used as an internal forward model for control.

So far only revolute joints were considered. This simplified the objective function as the joint positions could be expressed by scalars and the position error was given by the vector difference of the robot and simulator positions ($q_i^{\text{rob}} - q_i^{\text{sim}}$). However, more complex joint types, such as spherical joints, can be easily added to the objective function as long as the position error can be mapped to a scalar. If, for instance, quaternions are used to represent the joint position of a spherical joint, then the quaternion distance can be used as error measure for the objective function.

Another question that might arise is how many equilibria are required in order to calibrate a musculoskeletal robot. Let us therefore consider again a revolute joint setup actuated by *m* straight-line muscles, each equipped with a SEE. The joint is in equilibrium when the muscle and the gravitational torque are equal and the joint velocity is zero:

$$\tau_G = \sum_{i=1}^m \tau_{M_i} \wedge \dot{q} = 0$$
(5.10)

From Chapter 2 we recall that the 3-dimensional muscle torque τ_M with respect to a reference point is given by:

$$\boldsymbol{\tau}_M = (\boldsymbol{p} \times \hat{f}) f \tag{5.11}$$

where *p* is a vector from the reference point to any point on the line of action \hat{f} and *f* is the magnitude of the force, which depends on the expansion l_s :

$$f = g(l_S) \tag{5.12}$$

Muscle force error and the objective function

Other joint types

Required number of reference equilibria

72 MODEL CALIBRATION

Hence, by substituting f with the robot force f^{rob} and solving the function g for one of the unknowns from Equation 5.11 yields a system of 3 + m equations per equilibrium (one for each of the three coordinates of the torque vector and one for f and each muscle). Therefore, the number p of required equilibria is given by:

$$p = \left\lceil \frac{o - m}{3} \right\rceil \tag{5.13}$$

where *o* is the number of unknowns.

Example As an example let us assume again a revolute joint actuated by m straight-line muscles each equipped with a linear spring as SEE. Then the force f of each muscle is given by:

$$f = k l_S = k (l_M - l_T)$$
(5.14)

where *k* is the spring stiffness coefficient and l_M the muscle length. From Equation 4.16 we know that l_M of straight-line muscles is given by the Euclidean distance of the muscle origin and insertion point:

$$l_M = \sqrt{[p_O^x - p_I^x(q)]^2 + [p_O^y - p_I^y(q)]^2 + [p_O^z - p_I^z(q)]^2}$$
(5.15)

Note that it is assumed that the insertion point p_I depends on the joint position q. In the worst-case, the spring stiffness coefficient as well as the muscle origin and the insertion point are unknown. This yields a total of 7m unknowns (6 coordinates and k for each muscle). Hence, at maximum, 2m equilibria would be required for the calibration of this revolute joint example.

Statics calibration and friction The joints as well as the muscle routing of musculoskeletal robots are subject to dry friction. These dry friction properties affect the equilibrium and hence the objective function value as the acquired equilibrium becomes dependent on the initial condition (i. e. the starting posture that was used to acquire the equilibrium). To minimize the error introduced by these frictional properties, it is advisable to use identical initial conditions during the recording of the robot and the simulator equilibria. One simple way to achieve such identical initial conditions is to use another equilibrium, i. e. the resting posture of the robot, as starting condition.

Selection of calibration parameters The physics-based models of musculoskeletal robots comprise a huge parameters a parameters. For instance, the straight-line muscle model of a single Anthrob muscle already consists of 32 parameters. Hence, it is important to select a proper subset of the model parameters as object parameters for the calibration. Normally, parameters that are identified experimentally, that are available in data sheets or that are possible to extract from Computer-Aided Design (CAD) models should not be the first choice for the calibration. Furthermore, by analyzing the parameter set of the models we find that there exists a subset which only affects the dynamics of the robot, such as joint viscous friction coefficients. These parameters have no effect on the equilibrium of the robot and therefore should not be included in the statics calibration procedure. However, these parameters can be calibrated in a second calibration iteration as described in the following section.

5.3 DYNAMICS CALIBRATION

Let us consider the joint position and muscle force trajectories of a single revolute joint actuated by a single muscle as shown in Figure 5.4. Sampling these continuous trajectories at equidistant intervals results in a discrete representation of the trajectories. If we now compute and add the position and force errors of the model for each of the samples we obtain an objective function which can be used to guide the calibration of the model dynamics. Formally, this can be expressed as follows.

Let us consider again a robot comprising only revolute joints and let *T* be a sequence of *k* trajectory samples *S*:

$$T := \{S_1, \dots, S_k\}$$
(5.16)

Furthermore, let each sample be given by a quadruple of the form:

$$S_i := (t_i, l_{T_i}, f_i, q_i) \quad i \in \{1, \dots, k\}$$
(5.17)

where t_i , l_{T_i} , f_i and q_i are the time, the tendon lengths, the forces and the joint positions of sample *i*, respectively. Then the simulationreality gap of the physics-based model for a single trajectory can be quantified by: (i) recording the trajectory T^{rob} for the robot, (ii) replaying the trajectory in the simulation by setting the robot tendon lengths $l_{T_i}^{\text{rob}}$ within the simulator to record the corresponding simulator trajectory T^{sim} and (iii) compute the joint position and muscle force error for each pair of robot and simulator samples $(S_i^{\text{rob}}, S_i^{\text{sim}})$ similar to the objective function of the statics calibration. Hence, the fitness $\mathcal{F}_T(\mathcal{I})$ of individual \mathcal{I} for a single trajectory is given by the arithmetic mean of the fitness of all samples:

$$\mathcal{F}_{T}(\mathcal{I}) = \left[\sum_{i=1}^{k} \kappa_{q} \, \frac{\|\boldsymbol{q}_{i}^{\text{rob}} - \boldsymbol{q}_{i}^{\text{sim}}\|}{n} + \kappa_{f} \, \frac{\|\boldsymbol{f}_{i}^{\text{rob}} - \boldsymbol{f}_{i}^{\text{sim}}\|}{m}\right] / k \qquad (5.18)$$

If we now consider a set of *p* trajectories for the calibration of the model dynamics, the objective function $\mathcal{F}_D(\mathcal{I})$ is given by:

$$\mathcal{F}_D(\mathcal{I}) = \left(\sum_{i=1}^p \mathcal{F}_T(\mathcal{I})\right) / p \tag{5.19}$$

Objective function

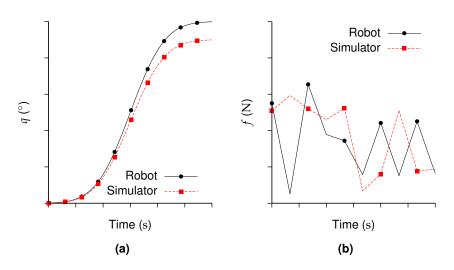


Figure 5.4: Example joint position and muscle force trajectories. The trajectory fitness of the simulator is computed by sampling the joint position and muscle force trajectories at equidistant intervals and calculating the mean error of all samples. (a) Joint position trajectory. (b) Muscle force trajectory.

```
Selection of object
                   The dynamics of the physics-based model are affected by all model
 parameters and
                   parameters. Hence, better dynamic calibration results are to be ex-
   initial values
                   pected if not only the dynamic parameters (e.g. joint viscous friction
                   coefficients) but also kinematic parameters are added to the set of ob-
                   ject parameters. Furthermore, the calibration result can be improved
                   by using the calibrated parameter values from the statics calibration
                   procedure as initial guess for the dynamics calibration. Such a se-
                   quential calibration process has been used in Chapter 8.
                   Equation 5.18 computes the fitness of a trajectory. However, if kine-
Fitness including
      equilibria
                   matic parameters, such as the coordinates of muscle via-point are
                   added to the set of object parameters, the mutation of these param-
                   eters not only affects the dynamics fitness but also the statics fitness
                   \mathcal{F}_S. In these cases, better calibration results can be obtained by consid-
                   ering the fitness of the starting \mathcal{F}_{S}^{s} and end equilibrium \mathcal{F}_{S}^{e} when cal-
                   culating the trajectory fitness. This yields the trajectory fitness \mathcal{F}'_T(\mathcal{I}):
```

$$\mathcal{F}_{T}'(\mathcal{I}) = \frac{\kappa_{s} \mathcal{F}_{S}^{s}(\mathcal{I}) + \kappa_{t} \mathcal{F}_{T}(\mathcal{I}) + \kappa_{e} \mathcal{F}_{S}^{e}(\mathcal{I})}{\kappa_{s} + \kappa_{t} + \kappa_{e}}$$
(5.20)

where κ_s , κ_t and κ_e are coefficients that define the contribution of the start equilibrium, the trajectory and the end equilibrium to the fitness.

6

THE MUSCULOSKELETAL ROBOT ANTHROB

In order to investigate the suitability of the physics-based simulation approach for simulating the dynamics of tendon-driven systems, the musculoskeletal robot Anthrob was used (see Figure 6.1). Anthrob, which was conceptualized at the end of the Eccerobot project, was developed together with Michael Jäntsch and the Swiss company awtec AG. The robot replicates the human upper limb (sometimes also referred to as upper extremity), which comprises the bones of the pectoral girdle, the (upper) arm, the forearm and the hand [41, 134]. This part of the human skeleton has been chosen as it includes structures that are typical for musculoskeletal robots and challenging for simulation and control, such as the spherical glenohumeral (shoulder) joint or biarticular muscles (muscles spanning two joints). All skeletal parts are designed in CAD tools to provide the required input data for the simulation model development and to facilitate the (re)-production of the robot (see Section 6.1). Whenever possible, 3D printing technologies based on Selective Laser Sintering (SLS) have been used for the production of the robot parts to evaluate cuttingedge technologies, such as tendon canals or solid-state joints, and to minimize the production costs. Furthermore, three types of muscle units have been developed to match the power requirements of the 13 skeletal muscles of the robot (see Section 6.2). Finally, proprioceptive and discriminative touch sensors have been included in the muscle units, in the joints and in the hand to provide sensory feedback for control and for the quantification of the simulation-reality gap (see Section 6.3).

6.1 SKELETON

The shoulder complex can be considered as one of the most complicated skeletal structures in humans. It consists of the glenohumeral joint (the shoulder joint) and the pectoral girdle [28]. Whereas the spherical glenohumeral joint enables the movement of the humerus with respect to the scapula in three mutually perpendicular axes (see Figure 6.2a), the pectoral girdle has two functions. First, it allows for scapula movements such as scapular elevation or depression [134]. Shoulder complex

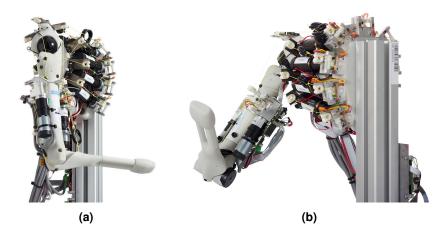


Figure 6.1: Photographs of the developed musculoskeletal robot Anthrob. The robot replicates the human upper limb and features 11 tendon-driven muscles (13 if the hand is attached, design: awtec AG, Zürich).

Second, it extends the range of humerus movements [119] (e.g. during an abduction of the arm of more than 90°). However, to reduce the complexity of the robot kinematics, only the glenohumeral joint was replicated. It consists of: (i) an immobile scapula, (ii) a glenoid cavity (or socket) and (iii) the humerus head (or ball) (see Figure 6.2b).

Elbow joint and In humans, the elbow joint is a compound joint comprising: (i) the forearm humeroradial joint (a spherical joint between the humerus and the head of the radius), (ii) the humeroulnar joint (a revolute joint connecting the humerus to the ulna) and (iii) the proximal radioulnar joint (a pivot joint between the radius and the ulnar) (see Figure 6.3a and [119, 134]). While the proximal radioulnar joint allows for pronation and suspination movements (the rotation of the radius around the ulnar), both the humeroradial and the humeroulnar joint contribute to the flexion and extension of the forearm [119]. Again, simplifications were made during the robot construction to imitate this complex compound joint. First of all, the ulnar and radius of the forearm were implemented as a single bone, which reduced the set of possible movements of the elbow joint to flexion and extension. Secondly, the omission of the radius and ulnar rendered the proximal radio-ulnar as well as two degrees-of-freedom of the humeroradial joint useless. Therefore, the elbow joint was implemented as a standard revolute joint with a maximum movement range of about 135° as shown in Figure 6.3b. This range is comparable to reported human data where the movement range is approximately 145° with slight variations between females and males [119].

Human hand and Anthrob design simplifications Another extremely complex but versatile human skeletal structure is the hand. Anatomically, it consists of the phalanges and the carpal and metacarpal bones and it is connected to the forearm via the dis-

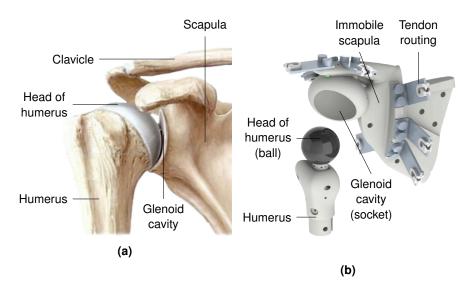


Figure 6.2: Human and Anthrob glenohumeral (shoulder) joint. (a) Human (licenced and taken from [134]), (b) Anthrob (design: awtec AG, Zürich)

tal radioulnar and the radiocarpal joint (see Figure 6.4a and [134]). This complicated skeletal structure allows for four primary types of grip: (i) the pinch or precision grip, (ii) the power grip, (iii) the key grip and (iv) the hook grip [134]. However, for the purpose of investigating the performance of robot controllers during manipulation tasks, the power grip is sufficient which yielded some design simplifications. First, the distal radioulnar and radiocarpal joints as well as the corresponding muscles were omitted and the hand was connected to the forearm via a passive, adjustable but lockable joint. This way, the posture of the hand does not change during manipulations but, if required, can be altered to match the requirements of a specific experiment. Secondly, the midcarpal joints, the joints between the proximal and distal rows of carpal bones, as well as the carpometacarpal joints, the joints between the carpal and metacarpal bones, of the index, middle, ring and little finger were not required.

The developed robot hand is shown in Figure 6.4b, 6.4c, and 6.4d. It includes the five metacarpals as well as the five digits with their proximal, medial and distal phalanxes. The metacarpophalangeal as well as the interphalangeal joints are implemented as solid-state hinges. This was possible as the entire hand was produced by state-of-the art 3D printing technologies, namely SLS. In this type of joint, the movement of the two connected bones along the joint axis is enabled by a very thin layer of material—here a 0.3 mm layer of Polyamid 2200 (PA-2200) (see Figure 6.4d). This approach not only resulted in a very slim and lightweight design but also made it possible to produce the hand from only two parts: (i) the thumb and (ii) the remaining four digits including the metacarpal and carpal bones. Moreover, due to the used 3D printing production technique, it was possible to

Anthrob hand

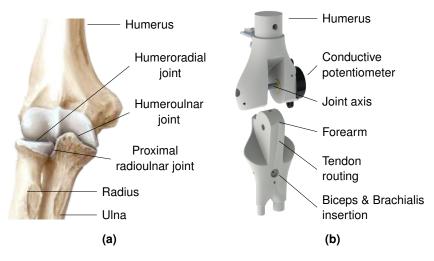


Figure 6.3: Human and Anthrob elbow joint. (a) Human (licenced and taken from [134]). (b) Anthrob (design: awtec AG, Zürich)

include tendon canals and ducts for the wires of discriminitive touch sensors (see Figure 6.4d and Section 6.3), which further reduced the production costs and simplified the hand assembly.

6.2 MUSCLES

In humans, the upper limb features more than 50 different muscles, of which some even constitute multiple heads. However, due to the skeletal simplifications made for the robot (see previous section), not all of these muscles need to be replicated in order to imitate human motions.

Muscles of the glenohumeral joint	For instance, many muscles of the shoulder complex are required for scapula or clavicle movements. However, due to the omission of these bones, only the muscles that are concerned with humeral movements are important for the robot. Furthermore, some muscle heads, such as the long and short head of the biceps brachii, can be merged with- out loosing a significant range of upper arm movements. Therefore, the glenohumeral joint of the robot is actuated by nine muscles with origin and insertion points similar to their human counterpart (see Figure 6.5, Table 6.1, and [134]).
Muscles of the elbow joint	In humans, three muscles contribute to the elbow joint flexion and extension: (i) the brachialis, (ii) the triceps brachii, which has both uni- and biarticular heads, and (iii) the biceps brachii, which only has biarticular heads [119, 134]. For the robot, all three muscles were imitated, but only the biceps brachii was implemented as a biarticular muscle (see Table 6.1). Again, the origin and insertion points were chosen to resemble human muscle functions.

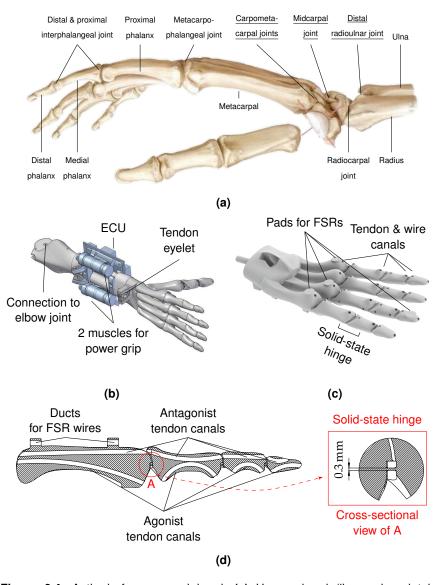


Figure 6.4: Anthrob forearm and hand. **(a)** Human hand (licenced and taken from [134]). Underlined parts are not included in the robot hand. **(b)** Anthrob forearm. **(c)** Anthrob hand. **(d)** Anthrob digit. Panels b-d: design: awtec AG, Zürich

The implementation of the robot hand muscles significantly differs from the glenohumeral and elbow joint muscles. The main reason is that the targeted power grip does not require each digit to be controlled individually. Therefore, each digit was equipped with an agonist and antagonist tendon that are contracted simultaneously by two muscles—one for the agonist and one for the antagonist tendons of all digits (see Figure 6.4d). However, to ensure that the hand adapts to the shape of the manipulated object during power grips, each of the 10 tendons featured a dedicated NBR ring as SEE (see Table 6.1).

A modular muscle unit, based on the principle of an electromagnetic series elastic actuator [122], has been developed for the muscles of the Anthrob robot. The muscle unit comprises a brushed DC motor,

Muscles of the hand

Muscle unit

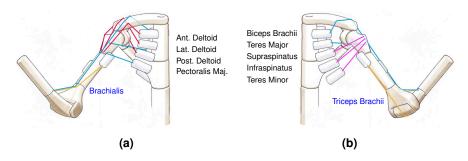


Figure 6.5: Anthrob glenohumeral and elbow joint muscles (black labels: muscle unit type A, blue label: muscle unit type B, see Table 6.2, illustrations by A. Jenter / awtec AG).

			Muscle	SEE			
	Muscle	X-articular	Unit	l_0	d	Quantity	
			Туре	(mm)	(mm)		
	Anterior Deltoid	uni	А	15	5	2	
nt	Lateral Deltoid	uni	А	15	5	2	
ioi '	Posterior Deltoid	uni	А	15	5	2	
моф	Pectoralis Major	uni	А	12	5	1	
Glenohumeral & elbow joint	Teres Major	uni	А	12	5	1	
	Teres Minor	uni	А	12	5	1	
	Supraspinatus	uni	А	12	5	1	
	Infraspinatus	uni	А	12	5	1	
	Biceps Brachii	bi	А	15	5	2	
	Triceps Brachii	uni	В	10	4	1	
	Brachialis	uni	В	15	4	1	
Hand	Agonist	multi	С	15	1	1*	
Ή	Antagonist	multi	С	15	1	1*	

Table 6.1: Properties of Anthrob muscles (* one for each of the five digits).

equipped with a gear and a spindle which is supported by a sliding bearing to prevent transmission failure (see Figure 6.6). As tendon, two types of high-performance kite-lines were used that are guided by ceramic eyelets originally developed for thread routing in the textile industry (see Table 6.2). Furthermore, each muscle was equipped with an NBR O-ring as SEE to mimic the viscoelastic characteristics of human muscles (see [151] and Figure 6.6b). Similar to the muscle units used in other musculoskeletal robots, the muscle is contracted by coiling the tendon on the motor spindle [91, 168]. Therefore, the maximum linear velocity and force applied by the tendon depend on the spindle radius as well as on the maximum velocity and torque of the gear output shaft which in turn is defined by the used actuator (DC motor and gear). Here, three different combinations are used

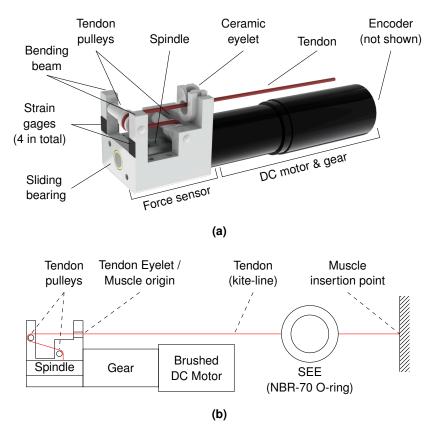


Figure 6.6: Anthrob muscle units. Three sizes of compact, replaceable muscle units have been developed for the Anthrob robot (type A, B and C—see also Table 6.2). (a) Muscle unit rendering (design: awtec AG, Zürich). (b) Muscle unit sketch.

to match the specific force requirements of the glenohumeral, elbow joint and hand muscles (type A, B and C, respectively—see Table 6.2).

All muscle units are controlled by distributed, Electronic Control Units (ECUs) developed by Jäntsch et al. [59], that are interfaced via a Controller Area Network (CAN). Each ECU is capable of controlling two muscle units and is equipped with a microcontroller, a CAN interface, motor drivers for two brushed DC motors, several Analog/Digital (A/D) converters for analog sensor connection and two integrated Hall-effect-based measurement devices in the motor loop for motor current feedback. To achieve a high degree of robustness during robot operation, each ECU was implemented as a fail-silent unit [126] with a firmware based on a Finite State Machine (FSM).

6.3 RECEPTORS

In humans, somatic sensation "arises from a variety of receptors distributed throughout the body" [64]. In robotics, particularly the receptors that contribute to proprioception ("the sense of static positions and movements of the limbs of the body" [64]) and to discriminative touch (the sensing and localization of touch) are important for conElectronic control units

Muscle	Actuator					Sensors	Tend	don
Unit	DC motor ¹	Gear^1	$\omega_{\rm nom}$	$\tau_{\rm nom}$	r _S	Position/	r_T	f _{max}
Туре			(rpm)	(N m)	(mm)	Force	(mm)	(N)
А	RE-25	GP22HP	64.38	1.98	6	•/•	0.5	2000
В	A-max 22	GP22HP	33.38	0.63	6	•/•	0.5	2000
С	RE-16	GP16A	22.24	0.57	4	•/-	0.3	400

Table 6.2: Anthrob muscle unit types.

Legend: ¹all by Maxon Motor, $\omega_{\text{nom}} \equiv$ nominal angular velocity, $\tau_{\text{nom}} \equiv$ nominal torque, $r_S \equiv$ spindle radius, $r_T \equiv$ tendon radius, $f_{\text{max}} \equiv$ maximum tendon force

trol and object manipulation. Therefore, the robot has been equipped with a variety of sensors to imitate these receptors.

In humans, three types of muscle and joint mechanoreceptors pro-Proprioceptive receptors vide proprioceptive feedback: (i) the muscle spindle receptors which provide muscle length, velocity and stretch feedback, (ii) the Golgi tendon organs that sense contractile forces exerted by a group of muscle fibers and (iii) the joint capsule mechanoreceptors that provide information about the current joint position [64]. Sensors to imitate these mechanoreceptors have also been included in the Anthrob robot. First of all, all muscle units were equipped with an encoder to measure the actuator position and velocity and with a Hall-effect-based measurement device, integrated in the ECUs, for actuator current feedback. Furthermore, all muscles of the glenohumeral and elbow joint (muscle unit type A and B, respectively) additionally featured a custom-built muscle force sensor¹ (see Figure 6.6). This sensor uses four strain gages that are attached to the two side arms of a bending beam. These four strain gages constitute a Wheatstone bridge which is used to measure the bending beam deflection and hence the muscle force. Moreover, to imitate the joint mechanoreceptors, the elbow joint was equipped with a conductive potentiometer. Unfortunately, similar sensors for the spherical glenohumeral joint are not available and optical substitutes are sometimes proposed. Urata et al., for instance, developed a spherical joint sensor based on a micro camera integrated in the joint for the musculoskeletal robot Kotaro [157]. However, this sensor is not commercially available. Instead, a high-speed, stereovision motion capture system was developed for Anthrob, that uses infrared light and retro-reflective markers mounted to the scapula and humerus of the robot to track the joint position and velocity of the glenohumeral joint. This motion capture system is described in detail in Appendix A.

¹ the integration of this sensor into the hand muscles (muscle unit type C) was not possible due to size constraints

In humans, the density of discriminative touch receptors is the "greatest on the hairless (glabrous) skin on the fingers, the palmar surface of the hand, the sole of the foot and the lips" [64]. Obviously, for object manipulation and identification, the finger and palmar surface receptors are particularly important. Therefore, the distal phalanx of the ring finger as well as three metacarpal heads of the robot hand were equipped with Force-Sensitive Resistors (FSRs) to measure the grip force during manipulations (see Figure 6.4c). *Discriminative touch receptors*

ANTHROB MODELING

In the previous chapter, the musculoskeletal robot Anthrob was introduced. In this chapter, the derivation of three physics-based models of this robot will now be presented. First, the skeleton model of these three models is presented in Section 7.1. Then, three different muscle as well as an ECU model are presented in Section 7.2 and Section 7.3. Finally, the processing times and the simulation-reality gap of all three models are analyzed in Section 7.4 and Section 7.5, respectively.

7.1 SKELETON MODELING

The Anthrob skeleton was modeled by three rigid bodies, called *base*, *humerus* and *forearm* (see Figure 7.1). While the humerus and the forearm were dynamic bodies (i. e. affected by forces), the base was set to be static. The mass and inertia properties of the bodies were derived from the available CAD data. Therefore, materials were assigned to each part of the CAD model and the resulting total mass, center of mass and moments of inertia were computed by SolidWorks. A summary of all parameters is provided in Appendix C. Note that no collision shapes were assigned to the rigid bodies as self-collisions are not relevant in the work-space of the robot and interaction scenarios were not considered. For the latter reason, the hand of the robot was also detached.

Two types of constraints are used within the Anthrob physics-based model: a ball-in-socket and a hinge constraint. While the former was used to model the 3 DoF glenohumeral joint of the robot, the latter was selected for the 1 DoF elbow joint. The Coulomb friction parameters of both joints (μ_s and μ_k , respectively) were set to zero to not bias the equilibrium joint positions, but were included later in the model calibration step (see Chapter 8). The viscous friction parameters, however, were non-zero and set to low values to stabilize the simulation. The Anthrob joint position sensors were simulated by *ideal* sensors (i.e. no noise or offset) and joint position as well as joint velocities were provided. A detailed overview of all constraint parameters is provided in Appendix C.

Rigid bodies

Constraints

7

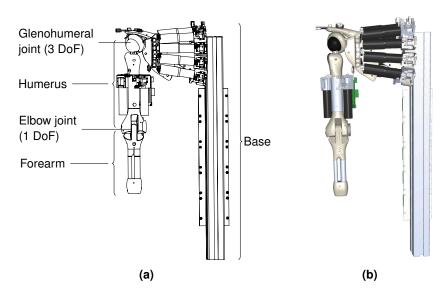


Figure 7.1: Anthrob physics-based skeleton model. **(a)** Anthrob skeleton kinematics. The Anthrob model comprises three rigid bodies (*Base, Humerus* and *Forearm*) as well as two constraints (a ball-in-socket constraint with 3 DoF for the spherical gleno-humeral and a hinge constraint with 1 DoF for the revolute elbow joint). A detailed summary of all skeleton model parameters is given in Appendix C. **(b)** Physics-based skeleton model rendered by the developed simulation software CALIPER.

7.2 MUSCLE MODELING

Models of the Anthrob muscles were derived and implemented for the developed simulator plugin of CALIPER. First, models of the Anthrob actuators (DC motor, gear and spindle) as well as three muscle kinematics models of increasing complexity were derived that are presented in Section 7.2.1 and Section 7.2.2, respectively. Moreover, two identification experiments were conducted to: (i) parametrize the via-point friction model presented in Section 4.4 and (ii) identify the stress/strain properties of the used SEEs made from NBR. The results of these experiments are summarized in Section 7.2.3 and Section 7.2.4.

7.2.1 Actuator Dynamics

For modeling the dynamics of the Anthrob actuators, the brushed DC motor model presented previously in Section 4.1.1 was used. All model parameters were taken from the datasheets of the motors, except for the spindle radius, which was measured manually and increased by the tendon radius to accommodate for the tendon thickness. All parameters are summarized in Table C.2. The model was numerically integrated using an explicit fourth order Runge-Kutta integrator with a step-size of 0.1 ms.

Table 7.1: Anthrob muscle kinematics models. Three muscle kinematics models of increasing complexity were developed for the physics-based Anthrob model. The three models include straight-line connections as well as cylindrical and mesh wrapping surfaces.

Kinematics	Shoulder Muscles			Elbow Muscles		
Model	Straight	Cylinders	Meshes	Straight	Cylinders	Meshes
Model I	•	-	-	•	-	_
Model II	•	•	-	•	•	-
Model III	•	-	•	•	•	-

7.2.2 Muscle Kinematics

Three types of muscle kinematics models of increasing complexity were derived for the Anthrob robot to compare and assess the resulting simulation-reality gap of each model in Section 7.5. These three models are (see also Table 7.1):

- 1. Muscle Kinematics Model I: This is the simplest model. All muscles are modeled by a sequence of via-points—one for each ceramic eyelet of the robot. Hence, no muscle wrapping on the skeleton was included.
- 2. Muscle Kinematics Model II: The muscle kinematics model I was extended by cylindrical wrapping surfaces to approximate the bone surface of the humerus for both, the glenohumeral and the elbow joint.
- 3. Muscle Kinematics Model III: The muscle kinematics model I was extended by mesh wrapping surfaces for the eight shoulder muscles and by cylindrical wrapping surfaces for the three elbow muscles (identical to model II). As wrapping meshes, the CAD data was used without modifications.

All parameters required for the models were taken from the CAD data. In case of the cylindrical wrapping surfaces, the cylinder radius was set to be equal to the Euclidean distance of the muscle insertion point on the humerus surface to the vertical axis of the humerus. A detailed summary of the muscle kinematics parameters for all three models is provided in Appendix C.

7.2.3 Nitrile Butadiene Rubber O-Rings

To mimic the viscoelastic properties of human muscles, each muscle unit of the Anthrob robot is equipped with an SEE made of NBR. Hence, an analysis of the stress/strain relationship of NBR was required to derive a simulation model for the physics-based simulation.

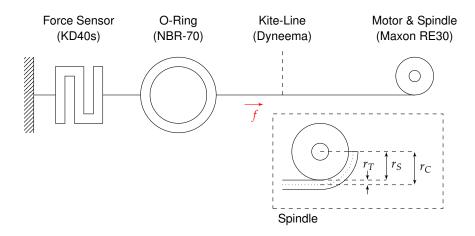


Figure 7.2: Experimental setup of the NBR identification. The tensile properties of the NBR O-rings listed in Table 7.2 have been identified with an experimental setup comprising: (i) a force sensor by ME Messsysteme (KD40s, $f_{Max} = 500$ N), (ii) a brushed DC motor by Maxon Motor (RE-30, 60 W), equipped with planetary gearhead GP32HP (reduction 51:1) and encoder MR256. The O-ring specimen was put under tensile load by coiling the Dyneema kite-line on the spindle. The target engineering strain ε_t was derived from the spindle position by taking the spindle and kite-line diameter into account (see bottom right). The resulting reference actuator position was controlled by a PD position controller with a proportional force feed-forward term implemented on a desktop computer.

The tensile properties of six NBR-70 O-rings with varying inner and Mechanical setup cross-section diameters were measured (see Table 7.2). The mechanical setup used for these experiments is shown in Figure 7.2. It consisted of (i) a force sensor by ME Messsysteme (KD40s) with a maximum load capability of $f_{\text{max}} = 500 \text{ N}$ and (ii) a brushed DC motor by Maxon Motor (RE-30, 60W) equipped with planetary gearhead GP32HP (reduction 51:1), encoder MR256 and a spindle with radius $r_{\rm S} = 1.2$ cm. The spindle was equipped with a bearing to protect the gearbox from the high radial loads occurring during the test runs. The O-ring specimen was attached to the force sensor and the motor spindle using Dyneema kite-line from Ockert (CLIMAX Protec) with a maximum load capability of $f_{max} = 2000 \text{ N}$ and a radius of $r_T = 0.5$ mm. However, the kite-line was not directly attached to the O-ring surface but through cable eyes to ensure a uniform distribution of the tensile force on the O-ring surface. This was necessary as preliminary tests had shown that the kite-line would otherwise damage the O-ring surface leading to a premature failure of the ring. The kite-line was connected to the cable eyes via a bowline knot.

Motor position control The O-ring specimen was put under tensile load by coiling the Dyneema kite-line on the motor spindle. The thereby applied engineering strain ε_e was controlled via a PD motor position controller with a superimposed proportional force feedback to ensure a good control performance under high loads. The controller was implemented on a desktop computer and executed with a frequency of 500 Hz. The

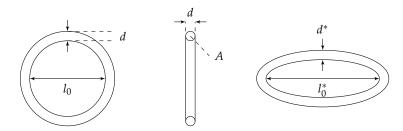


Figure 7.3: Overview of O-ring parameters. The dimensions of the O-ring are specified by the inner diameter l_0 and the cross section diameter *d* from which the cross section area *A* can be computed. When the ring is put under tension, it first deforms and becomes elliptic before the material is notably stretched. As a worst-case estimate for the inner diameter corresponding to the onset of the stretch, half the ring perimeter l_0^* was used which is equal to a engineering strain of ≈ 0.57 (see text).

custom-built ECUs (see [59]) were again used as interface boards to process the sensor signals and to drive the DC motor via a CAN bus running at 1 Mbit/s. The actuator position θ and velocity ω_G were derived from the motor encoder signal. The supply voltage of the analog force sensor was set to 10 V which yields a maximum output voltage of 5 mV at 500 N. Therefore, the force sensor output was amplified by a factor of 500 before it was converted into a digital signal using a 12 bit A/D converter to improve the resolution of the signal. Moreover, the digitized force was filtered using a Finite Impulse Response (FIR) filter of order 19. The gains of the PD controller were set to $K_P = 0.5 \text{ V/}^\circ$ for the proportional actuator position term, to $K_D = 0.01 \text{ V s/}^\circ$ for the derivative actuator velocity term and to $K_F = 0.02 \text{ V/N}$ for the proportional force term. The maximum output voltage of the controller was clipped at $\pm 8 \text{ V}$ to constrain ω_G .

The experimental protocol was split into a *preparation* and *recording* procedure. While the purpose of the preparation procedure was the tightening of the bowline knot and the pre-stretching of the Dyneema kite-line, the data acquired from the recording procedure was used for the model fitting and validation. Both procedures are described in the following paragraphs.

The preparation procedure proceeded as follows: First, the O-ring was attached to the force sensor and actuator spindle with a fresh pair of Dyneema kite-line. Subsequently, the kite-line was manually coiled until a minimum tension of $f_1 \approx 3$ N was reached and the corresponding motor position θ_1 and ring inner diameter l_1 were recorded (θ_1 and l_1 were used throughout the experiment to compute the target motor position θ_t corresponding to the target strain ε_t). In theory, the change in strain $\Delta \varepsilon$ induced by a change of the motor position $\Delta \theta$ should be equal to (see also bottom right panel in Figure 7.2):

$$\Delta \varepsilon = \frac{\Delta \theta}{l_0} \left(r_S + r_T \right) \tag{7.1}$$

Experimental protocol

Preparation procedure

	Ring Type #								
	1 2 3 4 5 6								
Supplier	HUG Industrietechnik								
Material	Nitrile butadiene rubber (NBR-70)								
<i>l</i> ₀ (m)	0.010	0.010	0.012	0.012	0.015	0.015			
<i>d</i> (m)	0.004	0.005	0.004	0.005	0.004	0.005			

Table 7.2: Tested O-ring types. The tensional properties of the following six different O-ring types with varying inner (l_0) and cross section (d) diameters were identified (see Figure 7.3).

However, this model is only valid under the assumption that the kiteline perfectly adapts to the spindle surface, a fact which was not confirmed by preliminary tests. Therefore, $r_S + r_T$ was experimentally determined and a value of 0.0129 m was used instead of the theoretically derived 0.0125 m. During the preparation procedure, the specimen was stretched 10 times for a duration of 10s to $\varepsilon_t = 1.1 \varepsilon_{\text{Max}}$ where ε_{Max} is the maximum strain level used during the recording procedure (see below). This approach had actually three effects: (i) the bowline knot was tightened, (ii) the kite-line, which when new exhibits a small stretch, was broken in and (iii) the ring was preconditioned to the strain levels used during the recording procedure to avoid the Mullins effect-a phenomenon of filled rubbers where the stress/strain relationship depends on the maximum load previously encountered [11]. To further compensate for the stretch of the kite-line and the tightening of the bowline knot, the motor position θ_1 was readjusted after each of the 10 preparation runs to match the initially recorded start force f_1 . The motor position θ_1 reached after the 10th preparation run was finally used to compute the target motor positions θ_t of the recording runs.

- *Recording procedure* After the preparation procedure was completed, each ring was tested in two recording runs. A single recording run was split into *n* relaxation tests at uniformly distributed and increasing target strains ε_t in the range of 0 and $\varepsilon_{Max} = 2.0$. Each of these *n* relaxation tests proceeded as follows: First, the specimen was stretched to the target strain ε_t for the recording time $t_R = 10$ s and the actuator position θ , force *f* and time *t* were recorded. After the expiration of t_R , the strain applied was removed for the duration of the recover time $t_C = 20$ s. Upon expiration of t_C , the specimen was stretched again to the next target strain until all *n* tests completed.
 - Model fitting &
validationThe experimentally observed strain/stress properties of the six tested
O-rings (see Table 7.2) were fit by two types of phenomenological
models: (i) a Zener model with only one Maxwell term (Model I) and
(ii) a Wiechert model with two Maxwell terms (Model II) (see also Sec-
tion 4.3.2). The fitting, which was based on the data of the 2nd record-

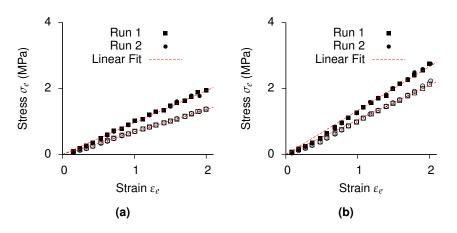


Figure 7.4: Initial and settled engineering stress vs. strain of both recording runs. The viscoelastic properties of NBR result in stress relaxations when a constant strain is applied. Therefore, the figures show both, the initial stress (filled symbols, averaged over the first 0.1 s after strain application) as well as the settled stress (unfilled symbols, averaged over the last 0.1 s of a 10 s recording period). (a) Ring type #2 ($l_0 = 10 \text{ mm}$, d = 5 mm). (b) Ring type #4 ($l_0 = 12 \text{ mm}$, d = 5 mm).

ing run, proceeded in two stages. In stage 1, the time-dependent relaxation behavior at a specific strain level was fit to obtain the spring moduli and relaxation time constants for that particular strain. In stage 2, the strain-dependent parameters of stage 1 were fit to obtain a model over the entire tested strain range. All fits are based on the engineering stress, which was computed by dividing the measured force by twice the cross section area of the ring *A* (see Figure 7.3). Furthermore, engineering strain levels ≤ 0.57 were ignored for the model fitting. The reason is that the ring first deforms to an ellipse before a notable stress of the material occurs. As a worst-case estimate for the inner diameter corresponding to the onset of the stretch, half the ring perimeter l_0^* was used which is equal to an engineering strain of ≈ 0.57 (see Figure 7.3):

$$\frac{l_0^*}{l_0} - 1 = \frac{\pi \, l_0}{2 \, l_0} - 1 \approx 0.57 \tag{7.2}$$

The two models resulting from the fit procedure were subsequently validated against a 2nd ring of the same type.

Examples of the stress-strain curves of two O-ring types (type #2 and #4, respectively) are shown in Figure 7.4a and 7.4b. To visualize the time-dependence of the stress due to the relaxation, both figures show (i) the initial stress, which corresponds to the beginning of the relaxation phase (filled rectangles), and (ii) the settled stress which represents the declined stress value after the expiration of the recording time t_R (unfilled rectangles). However, for statistical significance, both values are averaged over a 0.1 s time interval. Furthermore, in both figures the results of the two recording runs are shown superimposed to demonstrate the repeatability of the experiments, even though only

Stress/strain curves

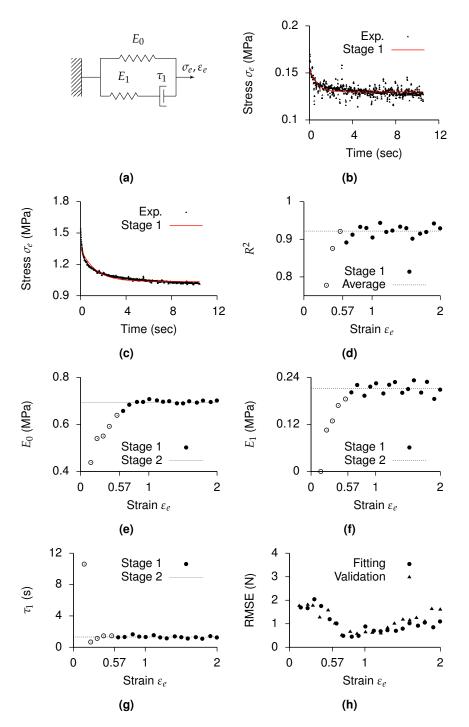


Figure 7.5: Example of SLS (or Zener) model fitting for ring type #2 ($l_0 = 0.010 \text{ m}$, d = 0.005 m). (a) Model and model parameters. (b) Relaxation at $\varepsilon_e = 0.24$. (c) Relaxation at $\varepsilon_e = 1.50$. (d) Coefficient of determination (R^2) for the stage 1 fit. (e) E_0 fit. (f) E_1 fit. (g) τ_1 fit. (h) RMSE of the model for the fitting and validation rings used. (Panels d-g) Unfilled data points have been neglected for the fitting (see text).

the 2nd run was used for the fit. The figures show that, within the experimentally tested strain range, the NBR O-rings exhibit relaxation

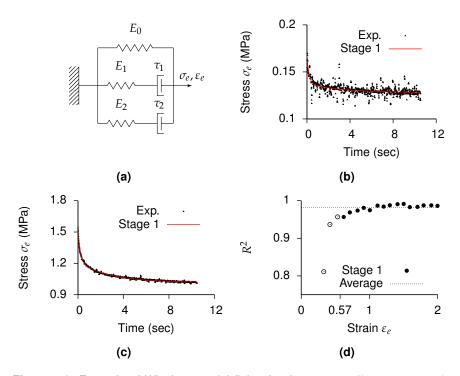


Figure 7.6: Example of Wiechert model fitting for ring type #2 ($l_0 = 0.010 \text{ m}$, d = 0.005 m)—Part A. (a) Model and model parameters. (b) Relaxation at $\varepsilon_e = 0.24$. (c) Relaxation at $\varepsilon_e = 1.50$. (d) Coefficient of determination (R^2) for the stage 1 fit. Unfilled data points have been neglected (see text).

and a linear stress-strain relationship, in accordance with the theory of linear viscoelasticity.

The first model (model 1) that was used to fit the experimental data was the three parameter Zener or standard linear solid (SLS) model (see Figure 4.13a and Figure 7.5a). The results of the model fitting for ring type #2 ($l_0 = 0.010 \text{ m}$, $r_R = 2.5 \text{ mm}$) are shown in Figure 7.5. The model is capable of faithfully reproducing the experimentally observed relaxation behavior, both at low and high strain levels as depicted in Figure 7.5b and 7.5c for $\varepsilon_e = 0.24$ and $\varepsilon_e = 1.50$, respectively. This observation was also confirmed by an average coefficient of determination (R^2) of 0.92 for all relaxations with strain level > 0.57 (see Figure 7.5d). However it should be mentioned that, due to single time constant of the Zener model, the largest fit error of the model was observed at the initial part of a relaxation (for $t_R \leq 0.2$ s, see Figure 7.5c). The strain-dependent spring moduli E_0 and E_1 as well as the relaxation time-constant τ_1 resulting from the stage 1 fit are shown in Figure 7.5e, 7.5f and 7.5g, respectively. As strain levels ≤ 0.57 are ignored for the stage 2 fit, all model parameters can be adequately approximated by a constant value, in accordance with the theory of linear viscoelasticity. This is confirmed by an extremely low RMSE for both the fitting and validation rings of 1.01 N and 1.13 N on average, respectively (see Figure 7.5h).

Model 1 (Zener) fit

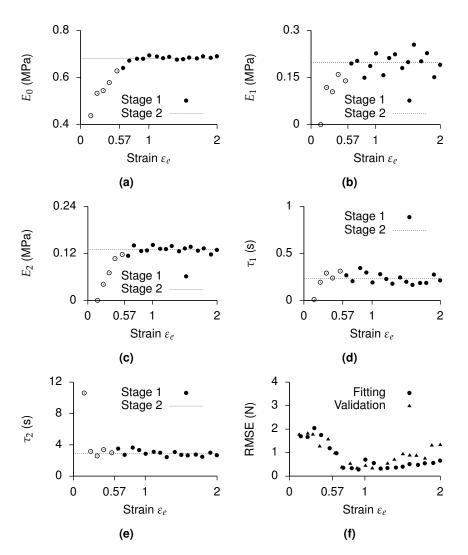


Figure 7.7: Example of Wiechert model fitting for ring type #2 ($l_0 = 0.010 \text{ m}$, d = 0.005 m)—Part B. (a) E_0 fit. (b) E_1 fit. (c) E_2 fit. (d) τ_1 fit. (e) τ_2 fit. (f) RMSE of the model for the fitting and validation rings used. (Panels a-e) Unfilled data points have been neglected (see text).

Model 2 (Wiechert) fit The second model (model 2) used for the fitting was a five parameter Wiechert model with two Maxwell terms (see Figure 4.14a and 7.6a). Similar to model 1, the Wiechert model was capable of capturing the dynamics of the relaxation experiments, both for low and high strain levels (see Figure 7.6b and 7.6c, respectively). However, in contrast to model 1, the Wiechert model provided a better fit for the beginning of the relaxations ($t_R \leq 0.2$ s), particularly for higher strain levels as shown in Figure 7.6c for $\varepsilon_e = 1.50$. This superiority is also reflected in higher coefficients of determination R^2 of the stage 1 fits, with an average of 0.98 (see Figure 7.6d). However, it can also be seen that the modulus E_0 of the model 2 fit is similar to the model 1 fit with 0.68 MPa on average compared to 0.69 MPa of model 1 (see Figure 7.5e and 7.7a). This was expected, as E_0 is defining the time-

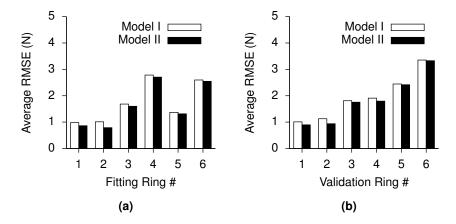


Figure 7.8: Summary of the NBR model identification experiment. **(a)** Average fitting RMSE. **(b)** Average validation RMSE. The more complex Wiechert model is slightly superior to the three parameter Zener model.

independent equilibrium stress σ_{∞} . The stage 1 results of the remaining model parameters E_1 , E_2 , τ_1 and τ_2 are shown in Figure 7.7b, 7.7c, 7.7d and 7.7e, respectively. Similar to model 1, all parameters can be approximated by a constant value if the strain levels ≤ 0.57 are ignored. This yields a final average RMSE of the Wiechert model of 0.79 N for the fitting and 0.94 N the validation ring (see Figure 7.7f).

The tensile and time-dependent stress/strain properties of six NBR O-rings with varying inner and cross-section diameters were experimentally analyzed to derive a computational model for the simulation of the O-ring dynamics. It was shown that NBR exhibits a repeatable, linear stress/strain relationship for the tested strain range. Furthermore, the parameters of two phenomenological models based on linear viscoelastic theory were derived to capture the experimental data: (i) a three parameter Zener or standard linear solid (SLS) model and (ii) a five parameter Wiechert model with two Maxwell terms. Here, it was shown, that both models are able to capture the time and strain-dependent dynamics of the NBR O-rings, with a slightly lower RMSE for the Wiechert model due its better fit of the initial part of the relaxations ($t_R \leq 0.2$ s). This observation is confirmed by the comparison of the model RMSE averaged over the entire strain range for both the fitting and validation rings (see Figure 7.8). However, as this difference will only have a minor impact on the overall model performance, the simpler SLS model was used for the simulation of the Anthrob model. It was numerically integrated using an explicit fourth-order Runge-Kutta integrator with a step-size of 0.1 ms. The identified model parameters of all tested O-ring types are listed in Appendix C.

Summary

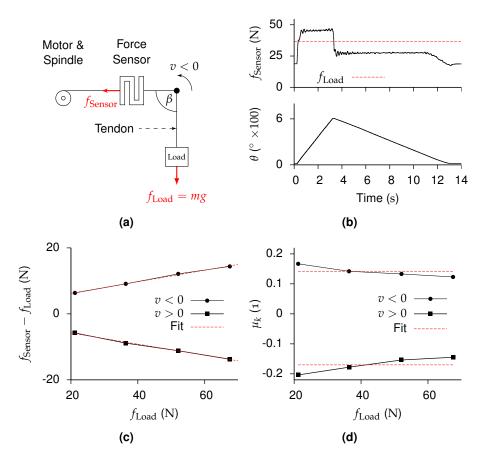


Figure 7.9: Identification of kinetic friction coefficient of ceramic eyelets. The kinetic friction coefficient of the ceramic eyelets was identified for the wrapping angle $\beta = 90^{\circ}$. (a) Experimental setup. (b) Raw data for $f_{\text{Load}} = 36.4 \text{ N}$. (c) Friction force vs. load force. (d) Derived kinetic friction coefficient μ_k using Equation 4.73 and Equation 4.74 $(|\overline{\mu_k}| = 0.141 \text{ if } v < 0 \text{ and } |\overline{\mu_k}| = 0.170 \text{ if } v > 0)$.

7.2.4 Ceramic Eyelets and Muscle Friction

The Anthrob robot features ceramic eyelets to route the tendons (see Chapter 6). These eyelets, which are normally used by the textile industry for thread routing, are subject to friction forces that are postulated to significantly influence the robot dynamics. Thus, their Coulomb friction properties were identified experimentally to parameterize the via-point friction model presented earlier in Section 4.4.

Experimental setup & The experimental setup is shown in Figure 7.9a. It consisted of a brushed DC motor by Maxon Motor (RE30, 60 W) equipped with planetary gearhead GP32HP (reduction 51:1), encoder MR256 and a spindle with radius $r_S = 1.2 \text{ cm}$ to pull and relax a load attached to a Dyneema kite-line (Ockert, CLIMAX Protec) routed through an eyelet. The force on the motor side of the eyelet f_{Sensor} was measured again by the KD40s load cell from ME Messsysteme (see Section 7.2.3). Both, the motor and the force sensor, were interfaced by one of the ECUs used for the control of the Anthrob robot (see Chapter 6). Four

types of loads, ranging from 21.2 N to 67.5 N, were tested. The mass of each load was determined and verified before the experiment using the KD40s force sensor and a scale with resolution 1g. As the friction force was postulated to increase for smaller wrapping angles β (see Section 4.4), the eyelet properties were identified for the smallest wrapping angle encountered in the Anthrob robot of $\beta = 90^{\circ}$. Measuring the maximum static friction force is inherently difficult as it would require to increase the reference voltage of the DC motor continuously until a constant, non-zero muscle velocity is reached. Hence, only the kinetic friction coefficient μ_k of the eyelets was identified during pulling (v > 0) and relaxing (v < 0)—the static friction coefficient μ_s will later be approximated by multiplication of μ_k with a factor (see below). Therefore, a reference voltage was set for the DC motor that guaranteed a constant velocity of the tendon on the eyelet and the force was recorded for approximately 2-3s. Subsequently, the raw force data was averaged over the periods with non-zero muscle velocity. A sample of the acquired data is shown in Figure 7.9b for $f_{\rm Load} = 36.4 \,\rm N.$

Results

The results of the experiment are summarized in Figure 7.9c and Figure 7.9d. It can be seen, that independent of the sign of v, the friction force $H_k = f_{\text{Sensor}} - f_{\text{Load}}$ increases linearly with the load force (see Figure 7.9c). Hence, the kinetic eyelet friction can be modeled by the Coulomb friction model presented in Section 4.4. Thus, to quantify μ_k for the Anthrob model, μ_k was computed for all eight data points by solving Equation 4.73 and Equation 4.74 for μ_k . The results are shown in Figure 7.9d. Not surprisingly, μ_k can be approximated by a constant. Moreover, and even more importantly, the magnitude of μ_k does not significantly change with the sign of v ($|\mu_k| = 0.141$ if v < 0 and $|\mu_k| = 0.170$ if v > 0). Hence, the kinetic friction force of the ceramic eyelets was modeled in the Anthrob simulation by the capstan friction model presented in Section 4.4 with $\mu_k = 0.156$. The static friction coefficient was set to $\mu_s = 0.195$, which corresponds to a μ_k/μ_s -ratio of 0.8.

7.3 ELECTRONIC CONTROL UNITS

The Anthrob muscle units are controlled by distributed, custom-built ECUs developed by Jäntsch et al. [59]. These ECUs are interfaced from the desktop computer via a CAN bus running at 1 Mbit/s and feature a microcontroller (STM32F from STMicroelectronics) to provide low-level control of the muscles. Currently, four different control modes are available: (i) actuator voltage, (ii) actuator position, (iii) actuator current and (iv) muscle force [58, 59, 168]. However, relevant to this work is only the actuator position control mode, which was used in this chapter to assess and in Chapter 8 to minimize the simulation-reality gap by means of model calibration. Hence, the actuator posi-

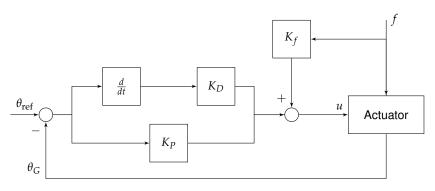


Figure 7.10: Actuator position controller. A proportional-derivative (PD) actuator position controller with a proportional force feed-forward compensation term was implemented for the Anthrob ECUs by Jäntsch [58]. This actuator position controller was ported to the CALIPER simulator to control the actuators of the physics-based model.

tion control mode had to be implemented for the Anthrob model to simulate the ECU behavior (see Section 7.3.1). Finally, independent of the active muscle control mode, reference voltages are computed and applied to the DC motor of the muscle via a full H bridge (Infineon BTS7810K). Hence, the input-output characteristics of the power circuit were identified in Section 7.3.2 and a model was derived.

7.3.1 Actuator Position Control

ECU A proportional-derivative (PD) actuator position controller with a proportional force feed-forward compensation term was implemented for the ECUs by Jäntsch (see [58] and Figure 7.10). The force feed-forward term was added to ensure a good control performance under high loads. The controller runs at a frequency of 1 kHz and both, the actuator position θ_G and velocity ω_G were derived from the motor encoder signal. The supply voltage of the analog force sensor was set to 10 V and the analog output was amplified before it was converted into a digital signal using a 12 bit A/D converter to improve the resolution of the signal. Moreover, the digitalized force was filtered using a FIR filter of order 60. Finally, the maximum output voltage of the controller was clipped at ± 12 V to constrain ω_G .

Simulator This actuator position controller was implemented for the simulation *implementation* of the physics-based Anthrob model. However, as the sensors were modeled as *ideal sensors*, i.e. no noise or offset, no pre-filtering of the force signal was included. Each controller was implemented as a thread and the control frequency was set to 500 Hz to reduce the computational load. This was possible without performance losses due the low controller gains used. All controller parameters are summarized in Table C.3.

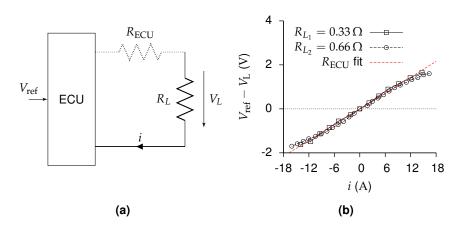


Figure 7.11: ECU power-circuit identification setup and results. The input-output characteristics of the power-circuit of the custom-built ECUs (see [59, 168]) used for the Anthrob robot were identified. (a) Experimental setup. The DC motor was emulated by a power resistor with resistance R_L inside the motor loop. (b) Experimental results. The measured voltage drop $(V_{\rm ref} - V_L)$ is quite significant but linearly dependent on the current *i*. Hence, the input-output characteristics of the ECUs were modeled by an additional resistor $R_{\rm ECU} = 0.12 \,\Omega$ in series with the DC motor.

7.3.2 Power Circuit Modeling

The actuator position controller computes reference voltages from reference positions (see previous section). In the ECU, these reference voltages are applied to the DC motor via a full H bridge which is controlled by a Pulse-Width Modulation (PWM) signal with a frequency of 50 kHz. Preliminary tests of the input-output characteristics of this power circuit revealed that the applied voltage was significantly lower than the commanded reference value. Hence, an identification experiment was conducted to first quantify and finally simulate this voltage drop within the physics-based model of the Anthrob robot.

The experimental setup is shown in Figure 7.11a. It consisted of an ECU connected to a load resistor with resistance R_L . This setup was preferred over a DC motor as it simplified the simulation of constant loads. Two levels of load resistance were tested: $R_{L_1} = 0.33 \Omega$ and $R_{L_2} = 0.66 \,\Omega$. As resistors, 100 W power resistors from ATE Electronics with a tolerance of 5% were used. The experiment proceeded as follows: (i) a reference voltage $V_{\rm ref}$ was commanded to the ECU, (ii) the voltage drop at the load resistor V_L as well as the current *i* was measured with an oscilloscope (Tektronix MSO 2024 equipped with current probe Tektronix A622). To ensure statistical significance, both quantities were averaged over 10 ms (500 PWM cycles). Even though the H bridges tolerate a peak current of 42 A, the current drawn during normal operation of a muscle unit is much less. For instance, the DC motor used in muscle unit type A (RE25 by Maxon Motor), is designed for a maximum continuous current of 2.33 A. Hence, the Experimental setup & protocol

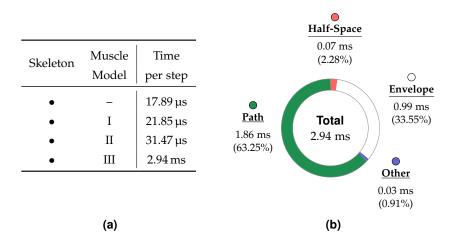


Figure 7.12: Benchmark results of physics-based Anthrob model. The execution time per simulation step of the implemented physics-based Anthrob model was analyzed. Therefore, the simulation was stepped 1000 times with a integration step size of 1 ms and the average runtime was computed (the timer resolution of 1 µs impeded a more detailed analysis). As the runtime is posture dependent, it should be noted that the skeleton model was in the equilibrium position (E_0 , see Section 7.5) and all muscles but the Anterior Deltoid were removed from the model. (a) Skeleton and muscle model runtimes. (b) Detailed analysis of the mesh wrapping surface runtime. (Muscle: Anterior Deltoid; Half-space: 748 points; Convex envelope: 149 points, 263 facets; Path: 34 points).

input-output characteristics of the ECU were not identified for the full H bridge current range of ± 42 A, but for approximately ± 16 A.

Results The results of the experiment are shown in Figure 7.11b. It can be seen that the voltage drop within the power circuit is significant (~ 1.6 V for i = 16 A) and linearly dependent on the current for both load resistance levels. Hence, the power circuit voltage drop was modeled by an additional resistor R_{ECU} with resistance 0.12 Ω within the motor loop as shown in Figure 7.11a and Figure 7.11b. It should be noted that this identified value is three times higher than the static drainsource on-state resistance (R_{DS-ON}) of the H bridge which is specified to be 40 m Ω . Hence, the H bridge is not the sole contributor to this voltage drop and other power circuit parts must be involved.

7.4 BENCHMARKS

The execution time of a single muscle was analyzed for the three implemented muscle kinematics models. Therefore, the simulation was stepped 1000 times with a numerical integration step-size of 1 ms and the average execution time per step was computed. The benchmark was performed on a desktop computer equipped with an Intel[®] CoreTM i7-860 quad-core processor¹. The results of the analysis are shown in Figure 7.12a. The baseline of all obtained values is the ex-

¹ the simulation was executed only on a single core

ecution time of the skeleton model alone, which requires 17.89 µs on average. If the straight-line muscle kinematics implementation of the Anterior Deltoid is added this value increases to 21.85 µs. Hence, the execution time of this straight-line muscle with 4 via-points, via-point friction, an actuator and a Zener model as SEE is equal to 3.96 µs (which is close to the resolution of the used timer of 1 µs). If a cylindrical wrapping surface is added to model the wrapping of the muscle on the humerus bone, the execution time per simulation step increases to 31.47 µs. Finally, if a mesh wrapping surface is substituted for the cylinder, the execution time was measured to be 2.94 ms on average-which is a hundred-fold increase compared to the cylindrical wrapping surface model. Of course, the execution time of this muscle model highly depends on the current posture of the robot and the density of the used mesh. However, the Anterior Deltoid has a long wrapping path on the humerus in the resting posture. Hence, the obtained results can be seen as almost the worst-case execution time of the mesh muscle model—at least for the Anthrob robot model. A detailed analysis of the contribution of the individual processing steps of the mesh wrapping surface to the average execution time is given in Figure 7.12b.

7.5 SIMULATION-REALITY GAP ANALYSIS

In Section 7.2, three muscle kinematics models of the Anthrob robot were presented. These three muscle kinematics models are now combined with the skeleton model presented in Section 7.1 to form three complete Anthrob models. In this section, the accuracy of the resulting three models will now be assessed by analyzing the simulationreality gap for both, equilibria postures and joint trajectories. This assessment proceeds in two tiers. In tier 1, the forearm is detached and the simulation-reality gap of the spherical glenohumeral joint with its 8 muscles is analyzed. In tier 2, the forearm is reattached and the errors of the complete robot model are quantified. The results of this analysis are presented in the following sections.

7.5.1 Tier 1: Glenohumeral Joint Models

To assess the simulation-reality gap of the three muscle kinematics models for static glenohumeral joint positions, a set Q_G^{rob} of 17 equilibria $E_G^{\text{rob}} := (\theta, f, q)$ was recorded for the robot. Note that the tendon lengths l_T , previously introduced in Chapter 5 as input data for an equilibrium, was substituted by the actuator positions θ as the tendon length cannot be measured directly with the Anthrob sensors. All equilibria were recorded by employing a zero-gravity control based on a computed-force controller developed by Jäntsch et al. [58, 60]. This controller utilizes a quadratic program to minimize the total

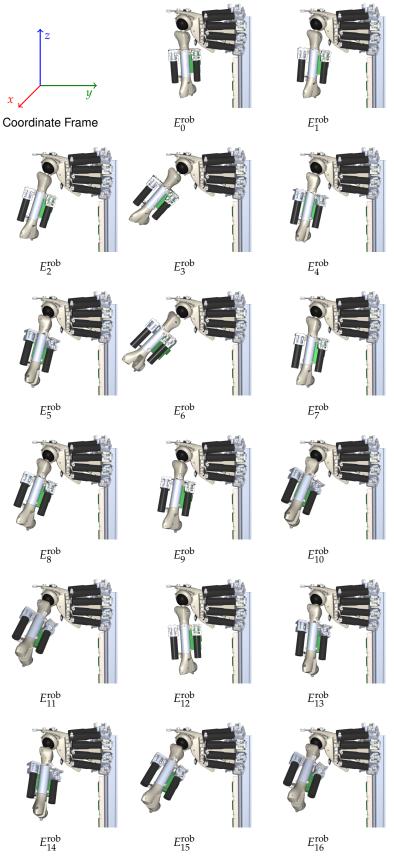


Figure 7.13: Glenohumeral joint equilibria.

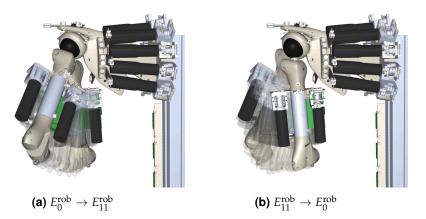


Figure 7.14: Example robot trajectory pair for equilibrium 11. A set T_G^{rob} of 32 trajectories was derived from the 17 equilibria shown in Figure 7.13 and recorded for the robot to quantify the dynamic errors of the models.

muscle forces required for the reference torque. Hence, muscle cocontractions were kept at a minimum. The recorded equilibria as well as the glenohumeral joint coordinate frame are shown in Figure 7.13. Furthermore, a set T_G^{rob} of 32 trajectories was derived from the 17 equilibria and recorded for the robot to quantify the dynamic errors of the models. Therefore, equilibrium 1 - 16 were acquired from equilibrium 0 (the resting posture) and vice versa—resulting in a pair of trajectories for each equilibrium. This acquisition was performed by setting the actuator positions θ_i^{rob} of equilibrium *i* for all muscles simultaneously as references for the actuator position controllers. Hence, the obtained trajectories are *emerging* from the musculo-skeletal interactions and no joint position feedback was involved. This is important as otherwise the controller would *dictate* the trajectory and the simulation-reality gap analysis would be flawed. One such trajectory pair for E_{11}^{rob} is shown in Figure 7.14. An overview of the mutual angular distances between the equilibria as well as a summary of the simulation-reality gap results is provided in Appendix C.

As error measures for the quantification of the simulation-reality gap, the joint position and the muscle force error were used—similar to the objective function of the calibration (see Chapter 5). In the case of the equilibria, these errors were computed by sampling the joint position and the muscle forces 0.5 s after all reference actuator positions were reached. For the trajectories, however, a similar approach is not possible. Here, the errors were measured for each trajectory sample *S* and the arithmetic mean was computed. The results of this analysis for each uncalibrated Anthrob model are presented in the following paragraphs. By *uncalibrated* it is meant that all model parameters were set manually, either from data sheets, identification experiments or from the available CAD data.

Simulation-reality gap error measures

104 ANTHROB MODELING

- Anthrob model I The equilibrium as well as dynamic joint position and muscle force errors of the uncalibrated Anthrob model I (straight-line muscle kinematics) are summarized in Figure 7.15 on page 105. In the equilibrium case, the mean joint position and muscle force error were 21.9° and 5.0 N, respectively. The highest joint position error was encountered for E_9 which was equal to 61.2° . The reason is that the muscle kinematics of the robot do not properly support rotations of the humerus around the *y*-axis of the glenohumeral joint. Therefore, this equilibrium was not only problematic for model I but for all models (see following paragraphs). The errors of the 32 trajectories are shown in two bottom panels of Figure 7.15. The average joint position error for all trajectories was 15.2°. For the *upward* case (from E_0 to E_x), the highest error was measured for E_{11} (26.8°). For the *downward* case, the worst trajectory was again involving E_9 (43.1°).
- Anthrob model II The performance of Anthrob model II (via-point muscles with cylindrical wrapping surfaces) is shown in Figure 7.16 on page 106. The mean glenohumeral joint position error $\|\overline{\Delta q_G}\|$ was reduced by almost 30% compared to model I to 15.6°. In contrast, the equilibrium muscle force error slightly increased by 8% to 5.4 N. This difference, however, is negligible. In the dynamic tests, model II also out-performed model I with respect to the mean joint position error (11.5° vs. 15.2°), whereas the muscle force error was almost identical (4.3 N vs. 4.4 N).
- Anthrob model III The final Anthrob model, model III, comprised via-points and mesh wrapping surfaces. The results of the simulation-reality gap analysis are summarized in Figure 7.17 on page 107. The mean joint position error of the glenohumeral joint was further reduced to 11.3° for the 17 equilibria. This is almost half the mean error of model I and more than 4° better than the result obtained for model II. A similar trend can be observed for the dynamic joint position errors. Here an average value of 8.7° was measured, which is 6.5° better than model I and 2.8° better than model II. The muscle force errors, however, are of similar magnitude than the ones obtained by model I and model II. For the equilibria, the average muscle force error was equal to 5.3 N, whereas it was 4.1 N for the tested trajectories.

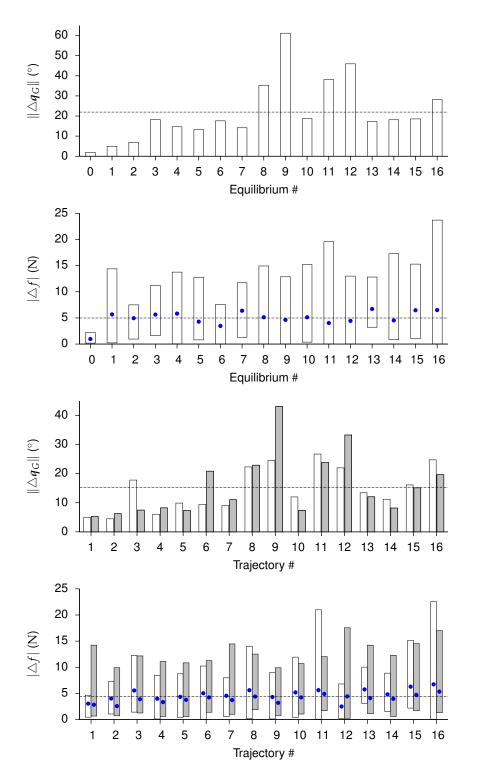


Figure 7.15: Tier 1 simulation-reality gap of Anthrob glenohumeral joint model I. Panel legend (from the top): (1) Glenohumeral joint position error per equilibrium. (2) Minimum, maximum and mean (•) muscle force errors per equilibrium. (3) Glenohumeral joint position error averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (4) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. All panels: (- -) Mean over all equilibria/trajectories.

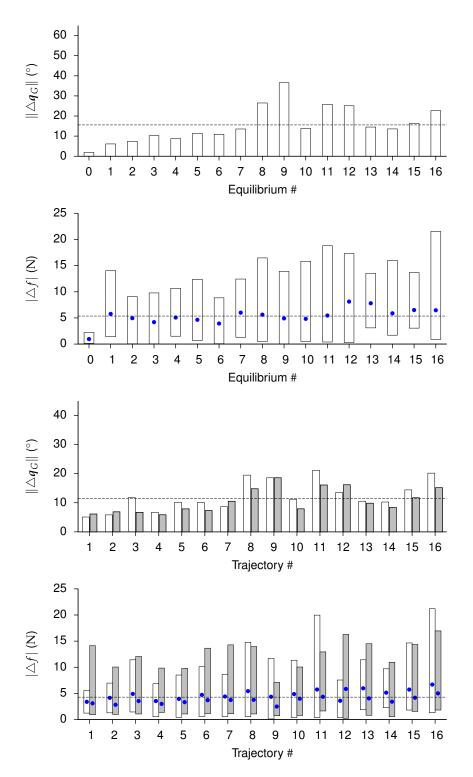


Figure 7.16: Tier 1 simulation-reality gap of Anthrob glenohumeral joint model II. Panel legend (from the top): (1) Glenohumeral joint position error per equilibrium. (2) Minimum, maximum and mean (•) muscle force errors per equilibrium. (3) Glenohumeral joint position error averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (4) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. All panels: (--) Mean over all equilibria/trajectories.

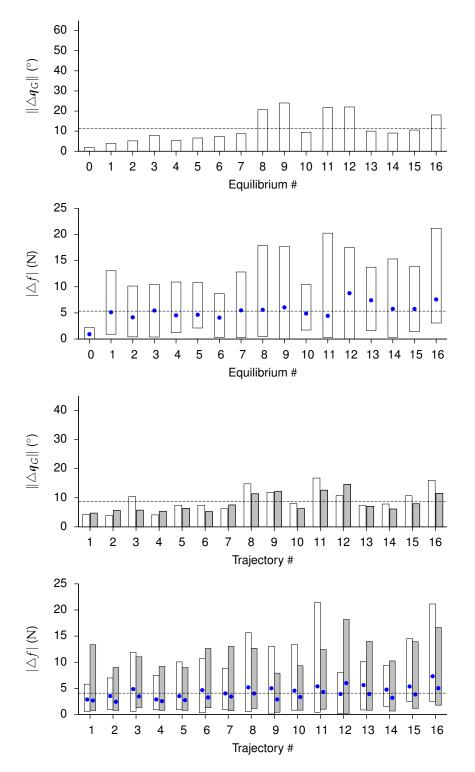


Figure 7.17: Tier 1 simulation-reality gap of Anthrob glenohumeral joint model III. Panel legend (from the top): (1) Glenohumeral joint position error per equilibrium. (2) Minimum, maximum and mean (•) muscle force errors per equilibrium. (3) Glenohumeral joint position error averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (4) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. All panels: (- -) Mean over all equilibria/trajectories.

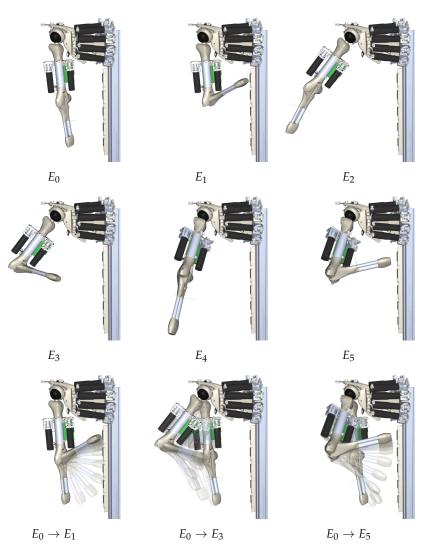


Figure 7.18: Anthrob equilibria and example trajectories. For the analysis and calibration of the complete Anthrob models, six equilibria were recorded. From these six equilibria, six trajectories have been derived ($E_0 \rightarrow E_1, E_1 \rightarrow E_0, E_0 \rightarrow E_3, E_3 \rightarrow E_0$, $E_0 \rightarrow E_5, E_5 \rightarrow E_0$). Three of these trajectories are shown in the bottom row.

7.5.2 Tier 2: Complete Robot Models

In this section, the simulation-reality gap of the three muscle kinematics models are presented for the complete Anthrob (humerus and forearm). Hence, another set Q_C^{rob} of six equilibria $E_C^{\text{rob}} := (\theta, f, q)$ was recorded—similar to the equilibria used for the glenohumeral joint analysis presented in the previous section. Furthermore, a corresponding set T_C^{rob} of six trajectories was again derived from the equilibria E_0 , E_3 and E_5 to quantify the dynamic errors of the models. The six equilibria as well as three example trajectories are shown in Figure 7.18. An overview of the mutual angular distances of the equilibria as well as a table summarizing the simulation-reality gap results can be found in Appendix C. The results of the simulation-reality gap analysis for Anthrob model I Anthrob model I are presented in Figure 7.19a on page 110. It can be seen that the mean equilibrium and dynamic joint position errors of the elbow joint are significantly lower than those encountered for the glenohumeral joint previously (4.4° and 5.1°, respectively). This was expected for two reasons: (i) the elbow joint has only one DoF and hence is less prone to erroneous muscle routing and (ii) muscle wrapping is not prominent for the elbow joint muscles. Hence, the straight-line muscle kinematics can provide a good approximation for the muscle path, particularly for elbow joint positions below 90°—above that angle, the Triceps muscle wraps around the robot skeleton. This is also visible in the results, as E_1 , E_3 and E_5 exhibit the largest errors.

The results of the simulation-reality gap analysis for Anthrob model II Anthrob model II are shown in Figure 7.19b. Here, in accordance with the tier 1 results, the glenohumeral joint position errors are lower than the ones obtained for muscle kinematics model I. However, the elbow joint position errors are slightly larger (0.7° for both, the statics and the dynamics analysis). Still, this was not expected as the cylindrical wrapping surface introduced for the elbow muscles should be capable of simulating the Triceps wrapping for joint positions above 90°. As we will see in Section 8.1, the reason for this is an erroneous manual parametrization of the cylinder radius. If the cylinder radius was calibrated, significantly better results could be obtained for the elbow joint position errors.

Finally, the results of Anthrob model III are presented in Figure 7.19c. Anthrob model III Again, in accordance with the tier 1 results, the more complex mesh wrapping surface model provided the lowest errors for the glenohumeral joint positions. However, as model III is identical to model II for the elbow joint muscles, no different results were obtained for this joint. The slight deviations in the elbow joint position error visible between Figure 7.19b and Figure 7.19c are only due to different glenohumeral joint positions which alter the effect of gravity on the forearm.

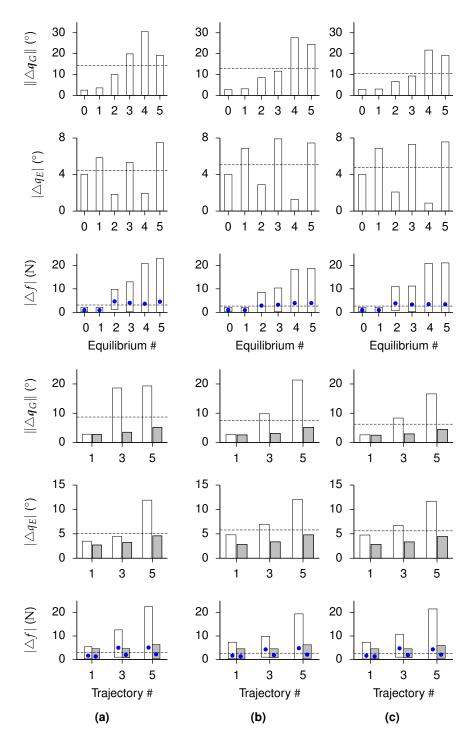


Figure 7.19: Tier 2 simulation-reality gap of Anthrob models. (a) Model I. (b) Model II. (c) Model III. Panel legend (from the top): (1) Glenohumeral joint position error per equilibrium. (2) Elbow joint position error per equilibrium. (3) Minimum, maximum and mean (•) muscle force errors per equilibrium. (4) Glenohumeral joint position error averaged over the entire trajectory ($\square E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (5) Elbow joint position error averaged over the entire trajectory. (6) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. All panels: (--) Mean over all equilibria/trajectories.

ANTHROB MODEL CALIBRATION

In Chapter 7, three Anthrob simulation models were presented and the simulation-reality gap of each model was analyzed. In this chapter, this gap is now minimized by applying the ES-based calibration procedure previously presented in Chapter 5 in Section 8.1. Once calibrated, the Anthrob models are validated in Section 8.2 to assess the quality of the calibration results. Finally, the chapter concludes by a discussion of the obtained results in Section 8.3.

8.1 CALIBRATION

The calibration proceeded in two tiers (see also Figure 8.1). In tier 1, the forearm of the robot was detached to calibrate the eight muscles of the spherical glenohumeral joint, whereas in tier 2 the forearm was re-attached and the 2 elbow joint muscles as well as the biarticular Biceps were calibrated for different glenohumeral joint positions. Both tiers were further subdivided into a statics and dynamics calibration stage. While in the former a set of equilibria Q was calibrated, the latter stage employed a set of reference trajectories T to minimize the dynamic errors of the three Anthrob models. Stage 1 was initialized by a manually parametrized ancestor individual \mathcal{I}_A , whereas the remaining three calibration stages always used the best individual of the preceding stage as initial condition. This results in a single object parameter set, i. e. the individual \mathcal{I}_{C} , for each Anthrob model. Furthermore, to reduce the risk that the calibration reached a local minimum, each stage was executed twice and the better run survived. The parent and offspring population sizes of all runs were set to $\mu = 6$ and $\lambda = 36$ to achieve a μ/λ -ratio of 1/6 which has been shown to provide a maximum rate of convergence [45, 163]. In all runs, the target fitness was set to zero (i.e. a perfect model with zero joint position and zero muscle force error). The coefficients κ_q and κ_f , that define the contribution of the two error measures to the objective function value, were determined in trial runs and set to $\kappa_q = 1$ and $\kappa_f = 10^{-4}$, respectively. Finally, a distinct set of model parameters was selected as object parameters for each Anthrob model. However, only model parameters that were not identified experimentally or that could not

Calibration procedure

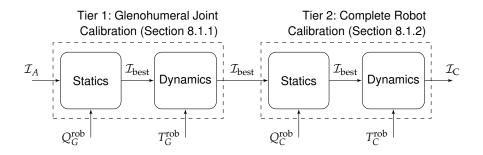


Figure 8.1: Anthrob calibration procedure. The three Anthrob models derived in Chapter 7 were calibrated in two tiers to obtain the individual \mathcal{I}_C for each model. In tier 1, the glenohumeral joint with its eight muscles was calibrated while the forearm was detached. In tier 2, the forearm was re-attached and the two elbow muscles (Brachialis and Triceps) as well as the bi-articular Biceps were calibrated. Both tiers consisted of a *statics* and *dynamics* calibration stage with disjunct equilibrium and trajectory input data (Q_x^{rob} and T_x^{rob} , respectively).

Table 8.1: Overview of calibration object parameters. Depending on the calibration stage and the Anthrob model a specific set of muscle and joint parameters was selected as calibration object parameters (a more detailed table can be found in Appendix D).

Stage	Anthrob	Muscle Parameters				Joint Parameters		
	Model	Via-Point	Actuator		Cylinder	Stribeck Friction		
		p	r_s	l_{T_0}	r	τ_s	$ au_k$	k_v
Statics	Model I	•	•	•	-	_	_	-
	Model II	•	•	•	•	-	_	_
	Model III	•	•	•	•*	-	_	-
Dynamics	Model I	•	•	•	_	•	•	•
	Model II	•	•	•	•	•	•	•
	Model III	•	•	•	•*	•	•	•

* only for the elbow muscles in tier 2

be derived from the CAD model were considered. This included for instance the actuator spindle radius, which is assumed to be constant in the simulation but might increase in reality when the tendon is coiled or the radius of cylindrical wrapping surfaces. A summary of the selected model parameters for each calibration stage and Anthrob model is given in Table 8.1. A more detailed overview of all parameters can be found in Appendix D.

8.1.1 Tier 1: Glenohumeral Joint Models

8.1.1.1 Stage 1: Statics Calibration

First the statics of the three Anthrob glenohumeral joint models were calibrated. Therefore, the 17 equilibria previously introduced in Sec-

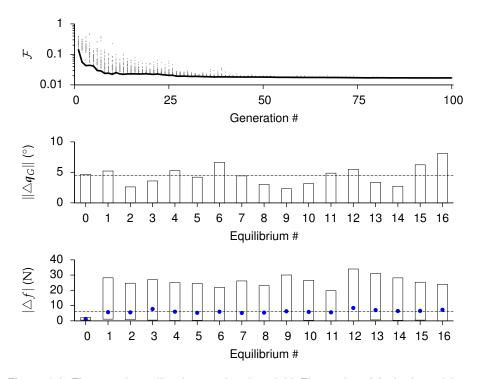


Figure 8.2: Tier 1 statics calibration results of model I. The statics of Anthrob model I (straight-line muscle kinematics) were calibrated using the 17 equilibria presented in Figure 7.13. Panel legend (from the top) (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Glenohumeral joint position error. (3) Minimum, maximum and mean (•) muscle force errors. (- -) Mean over all equilibria.

tion 7.5.1 for the simulation-reality gap analysis were used as input data for the calibration. A table summarizing the calibration results can be found in Appendix D.

For the calibration of model I (straight-line muscle kinematics), the following 5 muscle parameters were selected as object parameters for each of the 8 glenohumeral joint muscles: the initial tendon length l_{T_0} , the actuator spindle radius r_s and the x-y-z coordinates p of the last via-point before the muscle insertion point. For each of these 40 object parameters an initial mutation operator standard deviation σ as well as a minimum and maximum value range limit was defined (see Appendix D for a detailed summary). Moreover, the maximum number of generations for this run was set to $g_{max} = 100$. The results of this calibration stage are summarized in Figure 8.2. It can be seen that the calibration converges quickly during the first 10 generations, then stagnates for about 15 generations after which it continues to converge with a slower rate. The individual fitness was reduced by more than one order of magnitude from 0.219 to 0.017. This is also reflected in the glenohumeral joint position errors. Here, the mean error over all 17 equilibria was reduced by a factor of almost 5 from 21.9° to 4.5°. In contrast, the mean muscle force error over all equilibria and

Model I calibration

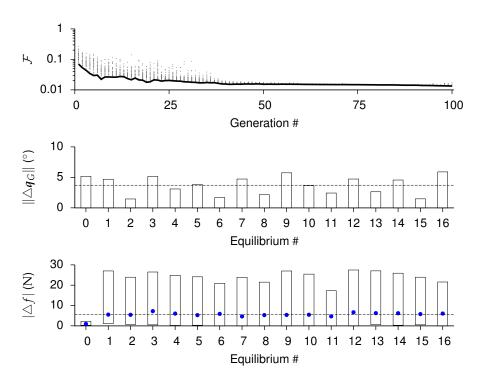


Figure 8.3: Tier 1 statics calibration results of model II. The statics of Anthrob model II (straight-line muscle kinematics with cylindrical wrapping surfaces) were calibrated using the 17 equilibria presented in Figure 7.13. Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Gleno-humeral joint position error. (3) Minimum, maximum and mean (•) muscle force errors. (- -) Mean over all equilibria.

muscles slightly increased by 1N from 5.0N to 6.0N. The main contributor to this force error is the high maximum force deviation of the Lateral Deltoid muscle of approximately 30 N for equilibrium E_1 to E_{16} . The Lateral Deltoid muscle is located laterally to the head of the humerus and hence contributes to almost all equilibria by lifting the humerus sideways (abduction). However, due to the small distance between the head of the humerus center of rotation and the final ceramic eyelet of the Lateral Deltoid on the Anthrob base, the lever arm of this muscle is small. This leads to high absolute forces and therefore high errors in the simulation. Furthermore, the wrapping angle of the Lateral Deltoid tendon on the eyelet is almost always close to 90° which results in high static friction forces on the eyelet surface during steady-states. However, these static friction forces can only be simulated by an approximation technique where the friction force depends on the muscle sliding velocity on the eyelet and if the velocity is zero, the friction force will be zero. Hence, the muscle force errors in the simulation are highest during steady-states and lower errors are to be expected for the calibration results of the model dynamics (see below).

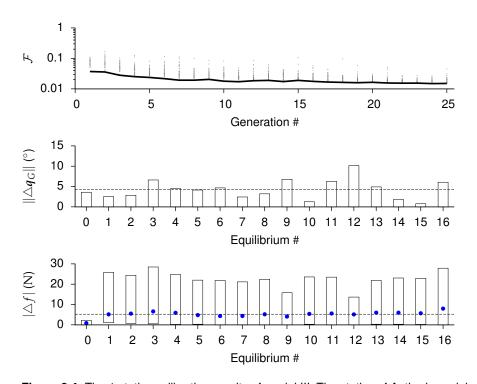


Figure 8.4: Tier 1 statics calibration results of model III. The statics of Anthrob model III (straight-line muscle kinematics with mesh wrapping surfaces) were calibrated using the 17 equilibria presented in Figure 7.13. Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Gleno-humeral joint position error. (3) Minimum, maximum and mean (•) muscle force errors. (- -) Mean over all equilibria.

For the calibration of model II (straight-line muscle kinematics extended by cylindrical wrapping surfaces), the cylinder radius r was added to the object parameter set. This resulted in a total of 48 object parameters. The maximum number of generations was set again to $g_{max} = 100$. The results of this calibration run are presented in Figure 8.3. Similar to model I, the fitness of the ES converges rapidly, eventually leading to a mean joint position error of 3.7° and a mean muscle force error of 5.5 N.

Finally, the statics of model III (straight-line muscle kinematics extended by mesh wrapping surfaces) were calibrated. Here, the identical object parameter set previously introduced for model I was used. However, as the simulation of the mesh wrapping surfaces is computationally expensive, the maximum number of generations was limited to $g_{\text{max}} = 25$. The mean joint position and muscle force errors of the best individual of this run were 4.3° and 5.2 N, respectively. The results are presented in Figure 8.4.

Model II calibration

116 ANTHROB MODEL CALIBRATION

8.1.1.2 Stage 2: Dynamics Calibration

The dynamics of the three Anthrob models were calibrated using the 32 trajectories introduced previously in Section 7.5. As object parameters, the same sets as for the statics calibration were used with the addition of the joint friction parameters (see Table 8.1). The muscle object parameters were initialized via the best individual of the preceding statics calibration stage, whereas the initial joint friction parameters were user-defined. An overview of all parameters is presented in Appendix D. As the calibration of the selected object parameter set would alter the previously obtained statics results, Equation 5.20 was used as objective function with the weight coefficients set to $\kappa_s = \kappa_t = \kappa_e = 1$ and the equilibrium errors were analyzed again for the best individual of the dynamics stage. The maximum number of generations was set to $g_{max} = 75$ for the model I and model II calibration runs and to 25 for the computationally more expensive model III. A table summarizing the calibration results can be found in Appendix D.

- *Model I* The dynamics calibration results of model I are summarized in Figure 8.5 on page 118. The obtained statics errors are almost identical to the stage 1 results with a mean joint position error of 4.6° and a mean muscle force error of 5.84 N. However, the trajectory errors are higher with an average joint position error of 9.9°. The mean muscle force error for the trajectory input data was equal to 4.0 N. But, as speculated previously, the maximum errors were lower than for the statics calibration stage due to the effect of the static eyelet friction modeling.
- Model II The dynamics calibration results of model II are shown in Figure 8.6 on page 119. While the mean muscle force errors of the equilibria and trajectories as well as the mean joint position error of the equilibria are similar in magnitude than the model I errors, the joint position errors of the trajectories are significantly lower. Here, an average value of 6.6° was obtained after 75 generations. Moreover, it can be seen that the trajectories that start at E_0 have almost consistently lower errors than the trajectories ending at E_0 . The reason for this is that the joint friction parameters were too low after calibration, resulting in temporary larger errors at the end of the trajectory as the humerus was passing through E_0 before it reached the steady-state (not shown). This may be compensated for by increasing the weight κ_t of the trajectory in the objective function.

The dynamics calibration results of model III are shown in Figure 8.7 Model III on page 120. Even though the maximum number of generations of this run was reduced from 75 to 25 generations to accommodate for the high computational load of the mesh wrapping surface model, the obtained results of the trajectory calibration are slightly better than for model II. For the glenohumeral joint position and muscle force an average error of 6.45° and 3.45 N was measured for the 32 calibration trajectories. Moreover, the observation made for model II that trajectories ending at E_0 have higher errors as trajectories starting at E_0 is not as prominent for model III. The reason is that the joint viscous friction coefficient of model III was almost double as high as for model II after calibration (0.113 N m s/rad vs. 0.057 N m s/rad). A slightly inverse situation was observed for the 17 equilibria. Here, model II slightly out-performed model III-at least in terms of the average joint position error (4.9° vs. 5.69° for model II and III, respectively). However, the difference is negligible and it is likely that model III would have reached considerable lower errors after a calibration run of 75 generations. Finally, the average muscle force error of model II and III deviate by 1.2N (3.7N for model II vs. 4.9N for model III).

As an example, Figure 8.8 on page 121 shows the errors of all three glenohumeral models after calibration for the trajectory $E_0 \rightarrow E_{11}$ (see also Figure 7.14a). This trajectory was selected as it exhibited the highest angular distance from the resting equilibrium E_0 (48.57°, see Appendix D). It can be seen that the joint position errors of model I and model II are higher during the dynamic phase of the movement (t < 1 s) and eventually reach a lower error when the target equilibrium is reached. Furthermore, the joint position error along the *z* axis is significantly higher for model I than for model II and III. The reason is that a rotation around the *z* axis requires the muscles to wrap around the surface of the humerus, which cannot be captured by the straight-line muscle kinematics model. Moreover, the detailed analysis of the muscle force errors confirms the statements made earlier that: (i) the Lateral Deltoid muscle is the main contributor to the average force error and (ii) that the Lateral Deltoid errors are the highest during steady-states as the muscle friction force model is ineffective in this case.

Example trajectory

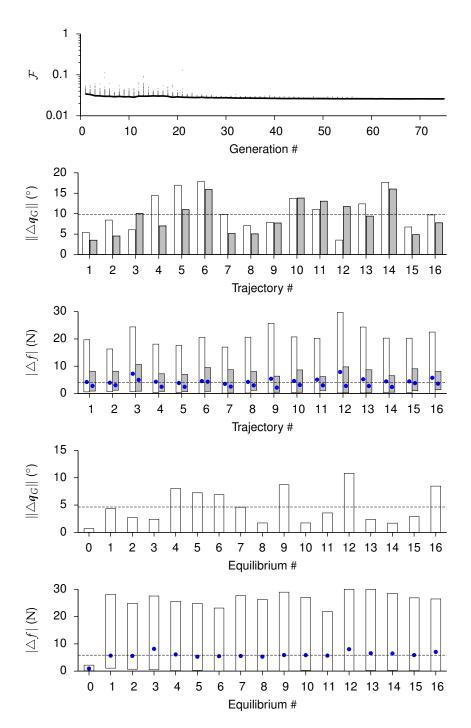


Figure 8.5: Tier 1 dynamics calibration results of model I. The dynamics of Anthrob model I (straight-line muscle kinematics) were calibrated using 32 trajectories (three top panels). As the dynamics calibration affected the equilibria errors (see text), these errors were reanalyzed (two bottom panels). Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Joint position error of the glenohumeral joint averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\Box E_x \rightarrow E_0$). (3) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. (4) Equilibrium joint position errors of the glenohumeral joint. (5) Equilibrium muscle force errors. All panels: (- -) Mean over all equilibria/trajectories.

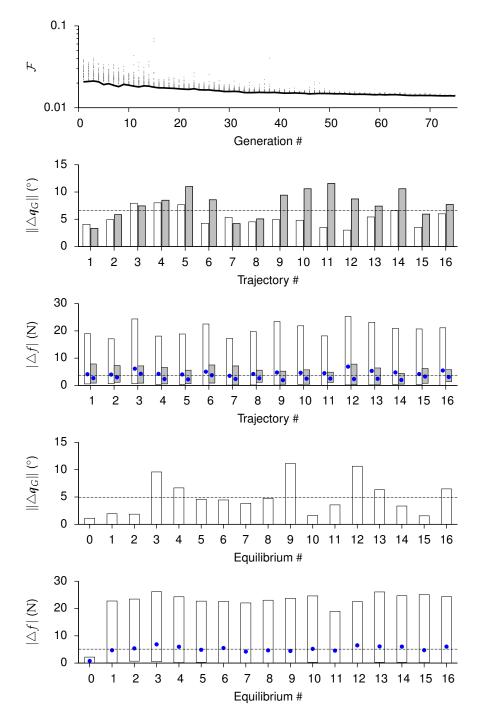


Figure 8.6: Tier 1 dynamics calibration results of model II. The dynamics of Anthrob model II (straight-line muscle kinematics with cylindrical wrapping surfaces) were calibrated using 32 trajectories (three top panels). As the dynamics calibration affected the equilibria errors (see text), these errors were reanalyzed (two bottom panels). Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Joint position error of the glenohumeral joint averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (3) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. (4) Equilibrium joint position errors of the glenohumeral joint. (5) Equilibrium muscle force errors. All panels: (- -) Mean over all equilibria/trajectories.

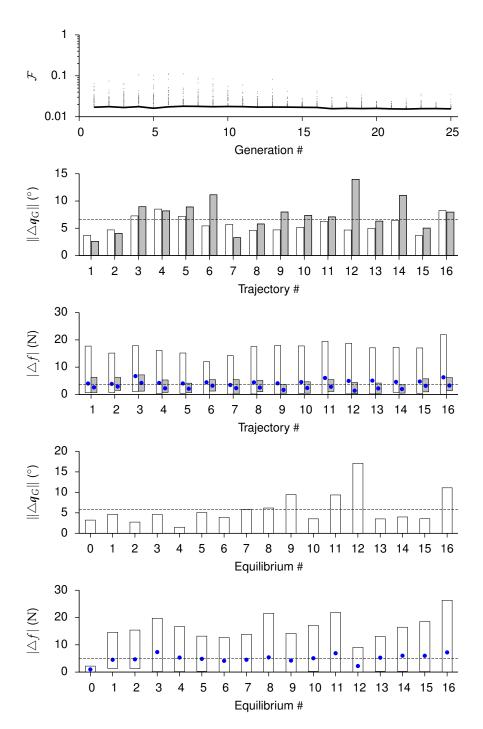


Figure 8.7: Tier 1 dynamics calibration results of model III. The dynamics of Anthrob model III (straight-line muscle kinematics with mesh wrapping surfaces) were calibrated using 32 trajectories (three top panels). As the dynamics calibration affected the equilibria errors (see text), these errors were reanalyzed (two bottom panels). Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Joint position error of the glenohumeral joint averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (3) Minimum, maximum and mean (•) muscle force errors averaged over the entire trajectory. (4) Equilibrium joint position errors of the glenohumeral joint. (5) Equilibrium muscle force errors. All panels: (- -) Mean over all equilibria/trajectories.

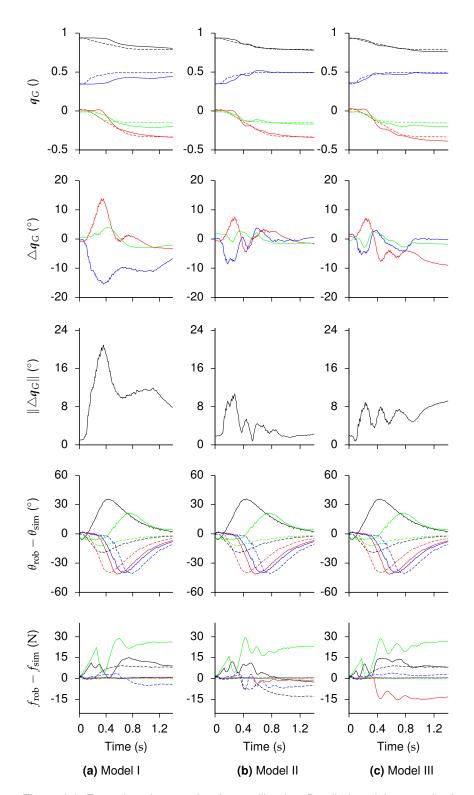


Figure 8.8: Example trajectory after tier 1 calibration. Detailed model errors of trajectory $E_0 \rightarrow E_{11}$. (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1) Anthrob (--) and simulation (—) glenohumeral joint quaternion (— w, -x, -y, -z); (2) Glenohumeral joint position error (— x, -y, -z); (3) Norm of glenohumeral joint position error; (4, 5) Actuator position and muscle force errors (— Anterior Deltoid, — Infraspinatus, — Lateral Deltoid, — Pectoralis Major, - Posterior Deltoid, - Supraspinatus, - Teres Major, - Teres Minor).

8.1.2 Tier 2: Complete Robot Models

In tier 2, the forearm of the robot was re-attached and the Biceps, the Brachialis and the Triceps muscle as well as the joint friction parameters of the elbow joint were calibrated again in a statics and dynamics calibration stage, respectively.

8.1.2.1 Stage 1: Statics Calibration

The statics of the complete Anthrob models were calibrated using the equilibria set Q_C^{rob} previously introduced for the analysis of the simulation-reality gap in Section 7.5.2. A table summarizing the calibration results can be found in Appendix D.

- Model I calibration The results of the model I calibration are summarized in Figure 8.9a on page 125. Similar to the tier 1 statics calibration, the initial tendon length l_{T_0} , the actuator spindle radius r_s and the x-y-z coordinates of the last via-point before the muscle insertion point were used as object parameters. However, only the newly introduced muscles (Biceps, Brachialis and Triceps) were calibrated and the best individual obtained for the dynamics calibration of tier 1 was used to parametrize the 8 glenohumeral joint muscles. The maximum number of generations was reduced to $g_{max} = 50$ due to the smaller set of 15 object parameters. It can be seen that the mean elbow position error is significantly lower than the glenohumeral joint error $(2.4^{\circ} \text{ vs. } 6.7^{\circ})$. This was expected and is in accordance with the results obtained for the simulation-reality gap analysis. Furthermore, by comparison with the statics results of tier 1 (see Figure 8.5), it becomes clear that the reattachment of the forearm had a huge impact on the glenohumeral joint position errors. While the maximum error was equal to 10.8° after the tier 1 calibration, it increased to 18.2° in tier 2. Moreover, the glenohumeral joint position errors in general exhibited a large jitter depending on the elbow joint position (see E_2 and E_3 as well as E_4 and E_5).
- *Model II calibration* The results of the model II calibration are summarized in Figure 8.9b on page 125. As this model used a cylindrical wrapping surface to model the muscle wrapping on the elbow joint for the Biceps, Brachialis and Triceps muscle, these cylinder radii were added to the set of object parameters. The maximum number of generations was set to $g_{max} = 50$. In contrast to the model I calibration results, model II neither did exhibit a jitter in the glenohumeral joint position error due to the elbow position nor did the glenohumeral joint position errors differ significantly from the tier 1 results. This shows the quality of the model which uses cylindrical wrapping surfaces to approximate the wrapping of the muscles on the humerus bone. Therefore, the model is able to better capture the muscle kinematics of the glenoh

humeral joint muscles which finally leads to lower joint position errors when the forearm is reattached. Furthermore, in comparison to the model I results, the average elbow joint error was reduced to 50% (1.2° vs. 2.4°). This shows that the cylindrical wrapping surfaces are capable of successfully approximating the muscle wrapping on the elbow joint skeleton. These calibration results also confirm the previous statement that the large simulation-reality gap of the elbow joint statics of model II were only due to an erroneous manual parametrization of the cylinder radius (see Section 7.5.2).

The results of the model III calibration are summarized in Figure 8.9c on page 125. The mean glenohumeral and elbow joint position error of this run was equal to 5.9° and 1.8° , respectively. The muscle force error was similar to the model I and model II results with an average value of 3.3 N. The maximum number of generations of this run was limited to $g_{\text{max}} = 10$ and it is likely that better results would have been obtained if $g_{\text{max}} = 50$ as for the model I and model II runs.

8.1.2.2 Stage 2: Dynamics Calibration

The dynamics of the complete Anthrob models were calibrated using the trajectory set T_C^{rob} previously introduced for the analysis of the simulation-reality gap in Section 7.5. A table summarizing the calibration results can be found in Appendix D.

In addition to the muscle parameters used for the statics calibration, the joint friction parameters of the elbow joint were added to the object parameter set. While the best individual of the previous statics calibration stage was used to initialize the muscle parameters, the initial joint parameters were manually configured. The maximum number of generations was further reduced to $g_{max} = 25$. The results of the tier 2 dynamics calibration of model I are summarized in Figure 8.10a and Figure 8.11a on page 126 and 127, respectively. It can be seen that the glenohumeral joint position errors are significantly lower for trajectories starting at E_0 (white boxes) compared to trajectories ending at E_0 (gray boxes). This phenomenon, which has been already encountered for the tier 1 dynamics of model II, is due to erroneous friction parameters after calibration and could be compensated for by increasing the weight of the trajectory κ_t in the objective function. But, in general, the tier 2 dynamics results are in line with the results of the preceding statics calibration stage.

Similar to the model I calibration, the best individual of the statics calibration stage was used to initialize the muscle parameters whereas the initial joint friction parameters were configured manually. Moreover, the maximum number of generations was set to $g_{max} = 25$. The results of the tier 2 dynamics calibration of model II are shown in Figure 8.10b and Figure 8.11b on page 126 and 127, respectively. In Model III calibration

Model I calibration

Model II calibration

accordance with the statics results, the glenohumeral joint position errors are significantly lower than the model I errors. However, the elbow joint position errors are of similar magnitude than the model I errors. This is surprising as the cylindrical wrapping surfaces introduced for the elbow muscles should provide a better approximation for the muscle kinematics than the straight-line muscles of model I.

- Model III calibration The results of the model III dynamics calibration are shown in Figure 8.10c and Figure 8.11c on page 126 and 127, respectively. The mean glenohumeral joint position error over all trajectories is equal to 5.0° which is slightly worse than the model II result (4.0°). In contrast, the mean elbow joint position error of model III is 0.8° better than the model II result (1.7° vs. 2.5°). The mean force errors are of similar magnitude for all three models with 2.9 N for model I, 2.7 N for model II and 2.6 N for model III.
 - *Example trajectory* The simulation-reality gap of all three calibrated models for the trajectory $E_0 \rightarrow E_5$ is shown in Figure 8.12 and Figure 8.13 on page 128 and 129, respectively. In accordance with the tier 1 results, the gleno-humeral joint position errors are similar in magnitude for the dynamic phase of the movement (t < 2 s) and are the highest around the *z*-axis. However, in the case of model I, the steady state error ($t \ge 2$ s) increases whereas it was decreasing during the tier 1 calibration. This is due to the re-attached forearm and the therefore increased mass of the entire arm which has a higher impact on the straight-line muscle kinematics model than on model II or III. In terms of the elbow joint position errors, all three models exhibited an extremely low error. However, the models with a cylindrical wrapping surface also outperformed the straight-line muscle kinematics in this setup even if not as clearly as for the glenohumeral joint position.

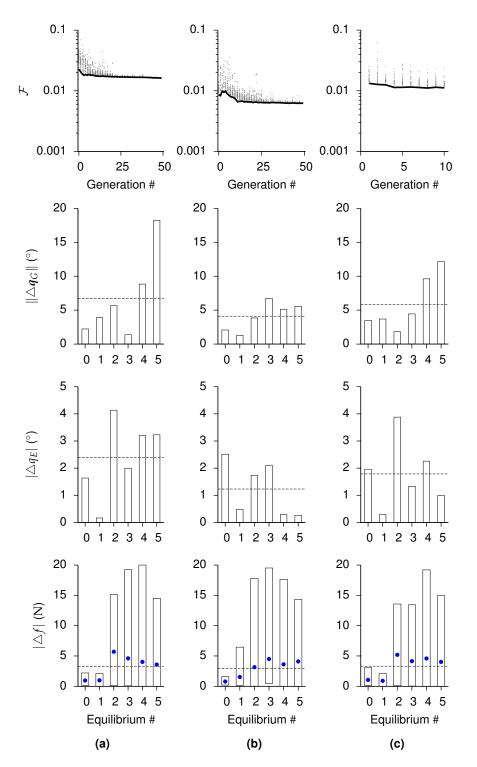


Figure 8.9: Tier 2 statics calibration results. The statics of the complete Anthrob models were calibrated using the equilibria set Q_C^{rob} . (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Glenohumeral joint position error. (3) Elbow joint position error. (4) Minimum, maximum and mean (•) muscle force errors. All panels: (--) Mean over all equilibria.

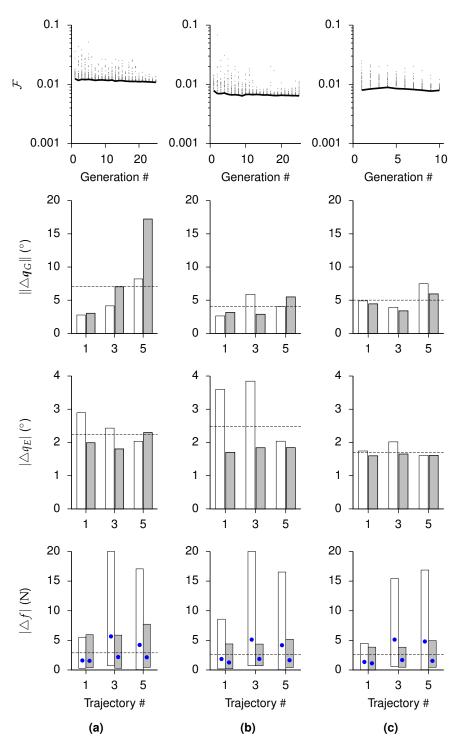


Figure 8.10: Tier 2 dynamics calibration results—part I. The dynamics of the complete Anthrob models were calibrated using the trajectory set T_C^{rob} . (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1) Objective function value \mathcal{F} (• Individuals, — Best individual of each generation). (2) Glenohumeral joint position error averaged over the entire trajectory ($\Box E_0 \rightarrow E_x$, $\blacksquare E_x \rightarrow E_0$). (3) Elbow joint position error averaged over the entire trajectory. (4) Minimum, maximum and mean (•) muscle force errors. All panels: (--) Mean over all trajectories.

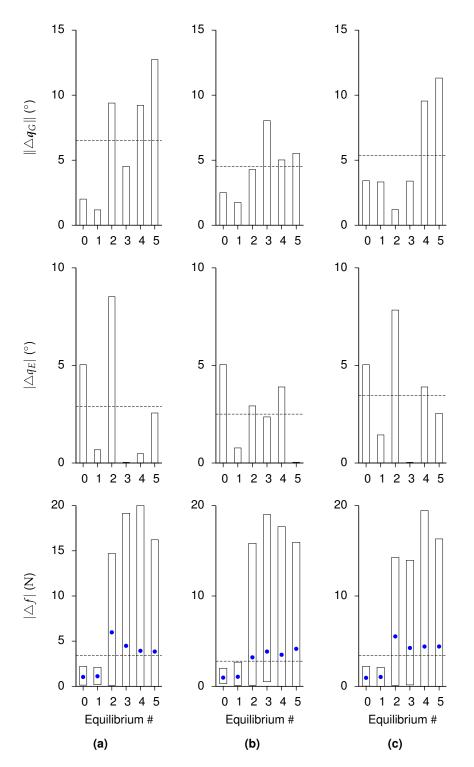


Figure 8.11: Tier 2 dynamics calibration results—part II. The dynamics of the complete Anthrob models were calibrated using the trajectory set T_C^{rob} . As the dynamics calibration affected the equilibria errors (see text), these errors were reanalyzed. (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1) Glenohumeral joint position error. (2) Elbow joint position error. (4) Minimum, maximum and mean (•) muscle force errors.

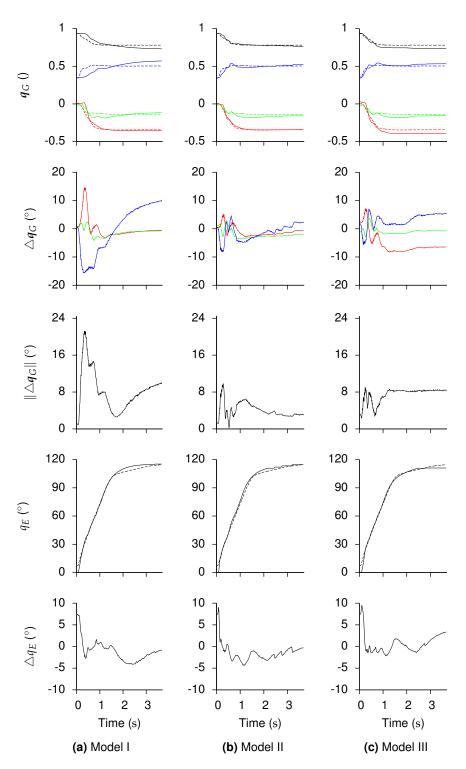


Figure 8.12: Example trajectory after tier 2 calibration—part I. Detailed model errors of trajectory $E_0 \rightarrow E_5$. (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1) Anthrob (--) and simulation (—) glenohumeral joint quaternion (— w, — x, — y, — z); (2) Glenohumeral joint position error (— x, — y, — z); (3) Norm of glenohumeral joint position error; (4) Anthrob (--) and simulation (—) elbow joint position. (5) Elbow joint position error.

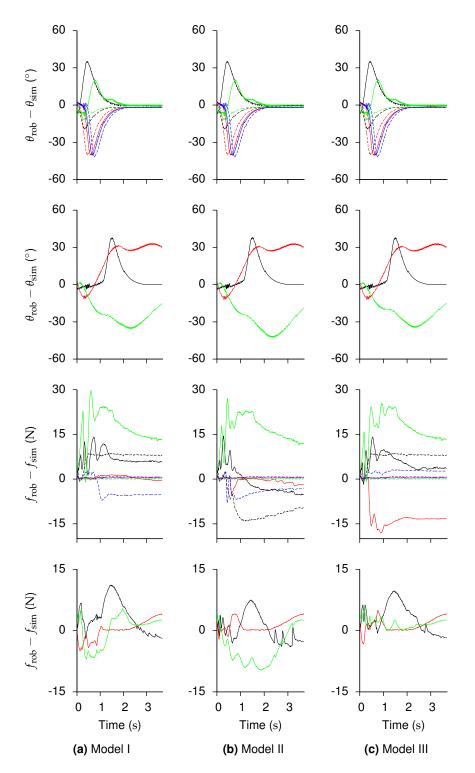


Figure 8.13: Example trajectory after tier 2 calibration—part II. Detailed model errors of trajectory $E_0 \rightarrow E_5$. (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1, 2) Actuator position errors. (3, 4) Muscle force errors. (1, 3) — Anterior Deltoid, — Infraspinatus, — Lateral Deltoid, — Pectoralis Major, -- Posterior Deltoid, - Supraspinatus, - Teres Major, - Teres Minor; (2, 4) — Biceps, — Brachialis, — Triceps.

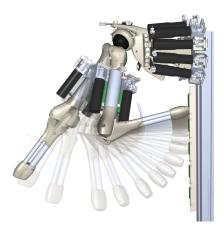


Figure 8.14: Validation trajectory. The calibrated Anthrob models were validated by analyzing the simulation-reality gap of a combined glenohumeral and elbow joint trajectory.

8.2 VALIDATION

In the preceding section, the calibration of the three Anthrob models was presented. This calibration resulted in three final parameter sets one for each model. In this section, these three parameter sets are now validated against a trajectory that was not used for the preceding calibration phase to assess the generalization quality of the models. Ideally, the simulation-reality gap of this validation trajectory would be of similar size as for the calibration data. Otherwise, the calibration would have *overfit* the calibration equilibria and trajectories—a problem also known from artificial neural networks where the network provides a perfect fit for the input data but only poor generalization [75].

- *Validation trajectory* The recorded validation trajectory is shown in Figure 8.14. It consisted of a combined glenohumeral and elbow joint movement with an angular distance of 50.7° for the glenohumeral and 115° for the elbow joint. The trajectory was again recorded via the zero-gravity controller developed by Jäntsch [58] to ensure that the start and end equilibrium are force-optimal (i. e. no co-contractions of antagonistic muscles).
 - Validation resultsThe validation results are summarized in Figure 8.15 and Figure 8.16,
respectively. Whereas the glenohumeral joint position error of model I
is significantly higher than for the calibration trajectories (peak error
 27.4°), this is not the case for model II and III. Here, the mean joint
position error of the glenohumeral joint is always below 10° and of
similar magnitude as observed for the calibration trajectories. Further-
more, similar to the calibration trajectories, the largest errors are still
observed for the z-axis—at least for model I and III. As for the elbow
joint, the error increases for angles below 90° and declines afterwards.
The highest peak error is measured for model I.

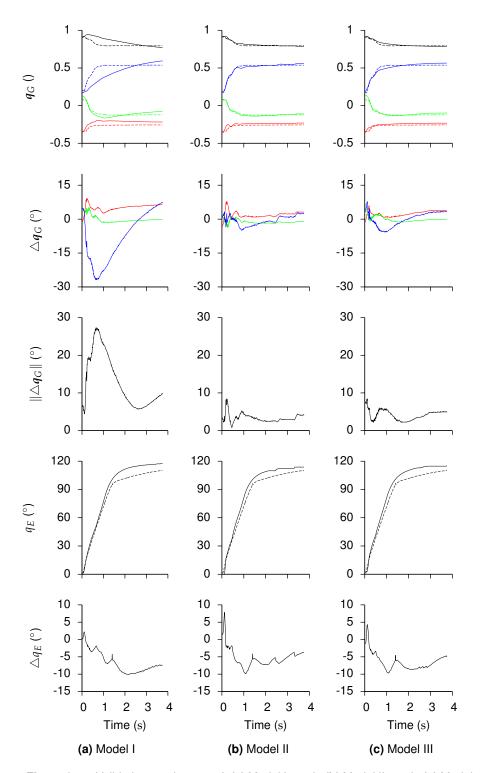


Figure 8.15: Validation results—part I. (a) Model I result. (b) Model II result. (c) Model III result. Panel legend (from the top): (1) Anthrob (--) and simulation (—) glenohumeral joint quaternion (— w, — x, — y, — z); (2) Glenohumeral joint position error (— x, — y, — z); (3) Norm of glenohumeral joint position error; (4) Anthrob (--) and simulation (—) elbow joint position. (5) Elbow joint position error.

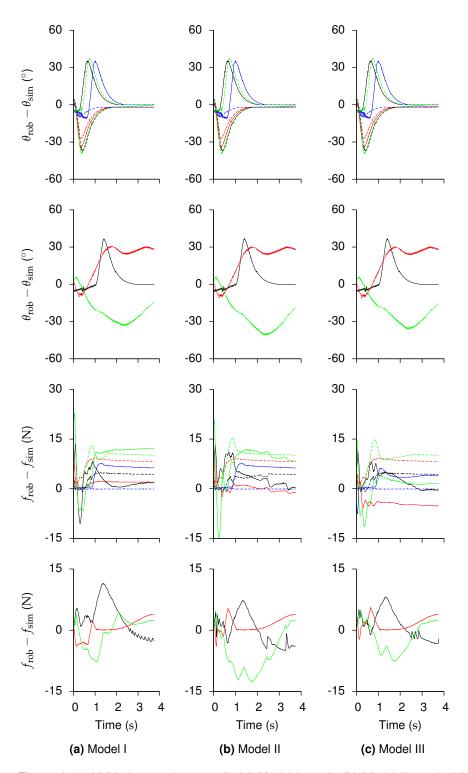


Figure 8.16: Validation results—part II. **(a)** Model I result. **(b)** Model II result. **(c)** Model III result. Panel legend (from the top): (1, 2) Actuator position errors. (3, 4) Muscle force errors. (1, 3) — Anterior Deltoid, — Infraspinatus, — Lateral Deltoid, — Pectoralis Major, - Posterior Deltoid, - Supraspinatus, - Teres Major, - Teres Minor; (2, 4) — Biceps, — Brachialis, — Triceps.

8.3 DISCUSSION

In this chapter, the statics and dynamics of three physics-based Anthrob models were calibrated successfully using the calibration procedure presented in Chapter 5. The three models differed in the modeling of the muscle kinematics, which mainly affect the glenohumeral joint position due to muscle wrapping. Thus, the simulation-reality gap reduction is most prominent for this joint as summarized in Figure 8.17 which compares the pre- and post-calibration errors of all three models for the equilibria set Q_C^{rob} and the derived trajectory set T_C^{rob} . Moreover, even though the pre-calibration error of the elbow joint was much less as for the glenohumeral joint, the simulationreality gap was also further reduced by the calibration procedure in most cases. However, the mean muscle force deviation remained almost constant for all models and calibration runs. The reason is that the muscle force error contributed only slightly to the objective function value due to the relative magnitude of the coefficients κ_q and κ_f .

For the calibration, only muscle and joint model parameters were considered that were not identified experimentally or that could not be derived from the CAD data. This approach resulted in a very limited object parameter set that was adjusted by the calibration procedure. Under the assumption that the used white-box models and the disjunct set of remaining model parameters is optimal (i.e. that the model parameters that are not affected by the calibration provide a perfect fit for their real counterpart), this limited object parameter set would result in an optimal simulation-reality gap (i.e. a simulationreality gap that cannot be further reduced). But this assumption cannot be made due to a variety of reasons, such as variations in material properties or physical effects that are neglected by the presented models (e.g. muscle wrapping friction). Hence, the selected object parameter set leads to a limited maximum reduction of the simulationreality gap and better results are to be expected if additional model parameters are considered as object parameters. This is confirmed by trial runs where previously identified model parameters, such as the spring modulus E_0 of the NBR model, were added to the object parameter set. This additional parameter significantly reduced the simulation-reality gap of all models, suggesting that additional identification experiments of sub-components of the robot would be required to further optimize the models.

The runtime of the calibration procedure depends on many parameters, such as on the input data or the exogenous strategy parameters of the ES. For the presented calibration runs the runtime varied between 352 min for the 100 generations of the tier 1 statics calibration of model I and 12 days for the 25 generations of the tier 1 dynamics Calibration & simulation-reality gap

Selection of object parameters

Calibration runtime

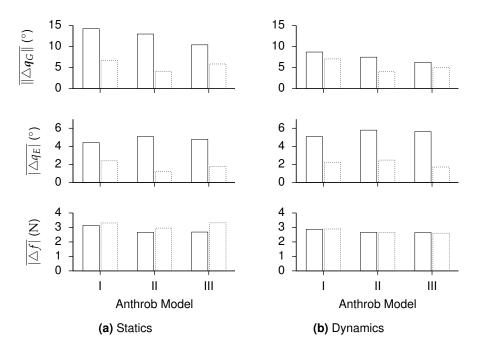


Figure 8.17: Anthrob model calibration results summary. The pre- and postcalibration joint position and muscle force errors are summarized for the three Anthrob models. (a) Static errors for the equilibria set Q_C^{rob} . (b) Dynamic errors for the trajectory set T_C^{rob} . Legend: (\Box) Pre-calibration. (\Box) Post-calibration.

calibration of model III. All runs were performed on an Intel[®] CoreTM i7-2600K quad-core processor, where multiple calibration stages were executed in parallel—each on a single core. It should be noted, that even though the runtime of a single run could be significantly improved by parallelizing the calculation of the muscle dynamics, the overall runtime of all calibration runs would have remained identical as the parallel execution of multiple runs on the same desktop computer would then not have been possible.

Actuator position errors & simulationreality gap The recorded trajectories were replayed in the simulation by setting the final recorded actuator position as reference for the actuator position controller. This approach resulted in maximum actuator position errors of approximately 30° (~ 3.4 mm, e.g. see top panels in Figure 8.16) and it seems to be a natural assumption that these errors contribute to the joint position and muscle force errors during movement. Therefore, trial runs were made where the gains of the controller were increased and each sampled actuator position was set as new reference value to ensure that the simulated actuator positions follow the recorded values as closely as possible (not shown). While this approach led to a significant reduction of the maximum actuator position errors (~ 5°), the joint position and muscle force errors did not considerably improve. Hence, this alternative approach for replaying the trajectory input data was not further considered. The ECUs used for the Anthrob robot do not feature an encoder interface. Therefore, the encoder ticks have to be counted in software which, due to the limited sampling rate available, leads to drifts in the actuator position for rapid movements. These drifts result in joint position errors and thus the Anthrob robot was re-calibrated during the recording of the calibration data as soon as the glenohumeral joint position error of the resting equilibrium E_0 was larger than $3^{\circ 1}$. This procedure induced deviations in the calibration input data due to the jitter resulting from the manual re-calibration. These deviations are for instance visible for E_5 and trajectory $E_0 \rightarrow E_5$ of Q_C and $T_{\rm C}$, respectively. Here the angular distance between the glenohumeral joint position of the equilibrium and the final sample of the trajectory is equal to 3.59°. This deviation is also reflected in the calibration results (e.g. the glenohumeral joint position error of E_5 in Figure 8.11c is $\sim 11.3^{\circ}$ whereas the final trajectory error is $\sim 8^{\circ}$ in Figure 8.12c). One option to solve this problem would have been to use the last sample of the trajectories as *target* equilibria. But this would only compensate for the deviations between the equilibrium/trajectory pairs. Deviations between pairs of equilibria would still persist. These could only be canceled out by the use of a hardware encoder interface which would allow the recording of all input data without the need of re-calibrations. Fortunately, the presented example can be seen as a worst-case example. Typically, these errors are below 1.5° .

For the validation of the calibrated Anthrob models, a single trajectory was used. Now, one might argue that one trajectory is not sufficient to assess the generalization capabilities of the models. This is not true for two reasons. First, the used validation trajectory has a duration of 3.75 s with a total of 372 samples. Hence, not a single robot posture but a sequence involving both robot joints and therefore the complete robot dynamics was used. Secondly, poor generalization normally is associated with the phenomenon known as over-fitting where the model provides a perfect fit for the training data instead of capturing the essential properties of the input data. This finally leads to large errors if untrained data sets are presented. Over-fitting, however, is a problem of black-box models, such as ANN, where a network structure (number of layers and neurons) is identified by supervised learning. In our case, also supervised learning was used but instead of black-box, white-box models were used. These white-box models were further constrained by value limits for each object parameter during calibration. Thus, it was not possible that the calibration resulted in unrealistic model parameter values and it is therefore unlikely that the model provides poor generalization—as confirmed by the low errors of the validation trajectory.

Anthrob repeatability

Validation trajectory

¹ the elbow joint muscles were not critical due to the lower controller gains which yield slower movement speeds

CONCLUSIONS

9.1 SUMMARY

Musculoskeletal robots, which belong to the super-class of tendondriven robots, exhibit a set of unique features such as spherical joints or muscles with Posture-Dependent Lever Arms (PDLAs). These features enable new applications for this class of robots, such as an artificial test-bed for the investigation of biologically inspired control strategies, or as service and rehabilitation robots, where the muscle compliance increases the safety of human-robot interactions. But, while being advantageous for many use-cases, these features also introduce new challenges for roboticists as outlined in the introduction. For instance, in the field of modeling and simulation the usage of muscles instead of torque actuators located within joints complicates the application of classical joint-space simulation approaches as an explicit mapping between the muscle forces and the joint torques is required. This mapping, however, is posture-dependent and therefore difficult to derive. Hence, the application of a different simulation approach for the class of musculoskeletal robots, namely physics-based simulations, was proposed.

Physics-based simulations, which originate from computer games and computer animations, operate in the body-space which means that body positions and orientations are integrated over time instead of the joint positions as in joint-space approaches. This difference simplifies the simulation of musculoskeletal robots as muscle forces can be applied directly to the rigid bodies of the skeleton without the need of force-to-torque conversions as in joint-space approaches. In this work, the applicability of this alternative simulation approach for the class of musculoskeletal robots was evaluated. Therefore, the modeling of skeletons by means of articulated rigid bodies was presented and it was shown how to extend the joint models of physics-engines to simulate joint friction. Subsequently, in order to actuate the passive skeletons, physics-based muscle models were derived. Here, particular emphasis was put on the muscle kinematics, which determine the lines of action of a muscle as well as the muscle length-two quantities that significantly influence the resulting robot dynamics. Problem summary

Physics-based modeling of musculoskeletal robots Thus, models of varying complexity were presented. These models ranged from simple straight-line connections to wrapping surfaces that employ geometric primitives, namely spheres and cylinders, to arbitrary meshes from which the roboticist can select the proper models for a particular robot and/or use-case to simulate the muscle path along the skeleton surface. Even though similar models already exist in the field of biomechanics, the models were ported to the physicsbased framework and were improved when possible. For instance, the sphere and cylinder wrapping surface models presented were extended to capture the transition from a positive to a negative wrapping direction as well as to support multiple wrapping revolutionstwo properties not supported by previous models. Furthermore, it was shown that the approximation of the geodesic path in the case of mesh wrapping surfaces, as suggested by previous biomechanical studies [23], should be avoided for physics-based simulations. The reason is that these approximations normally only provide good estimates for the path length but not for the path waypoints. These, however, are particularly important in the physics-based framework in order to compute the lines of action of the muscle. Moreover, it was shown how the presented muscle models can be extended to capture the effects of muscle friction forces which are postulated to have a significant impact on the dynamics of musculoskeletal robots [58]. Finally, a universal algorithm for the simulation of muscle dynamics within physics-engines was developed. This algorithm exploits the fact that the lumped effect of a wrapped muscle segment on the skeleton can always be simulated by two straight-line segments (one from the preceding via-point to the point where the muscle touches the surface and one from the point where the muscle leaves the surface to the succeeding via-point).

Model calibration
procedureThe parametric, physics-based white-box models of musculoskeletal
robots exhibit a huge parameter set and even though most of these
parameters can be derived from either data sheets, CAD models or
experiments, there will always be a simulation-reality gap. In order to
minimize this gap, a calibration procedure was developed. This pro-
cedure uses an Evolution Strategy (ES) with a Gaussian-based, non-
isotropic mutation operator to reduce the simulation-reality gap by
adjusting the model parameters. It is universal in the sense that it
can be applied to any real-valued model parameter and/or objective
function. As examples two objective functions were derived: one for
the calibration of the model statics and one for the model dynamics.
Both functions use the joint position and muscle force errors as fitness
measure to guide the exploration of the search space by the ES.

Both, the developed physics-based models and the calibration procedure were evaluated on a physical robot. Therefore, the musculoskeletal robot Anthrob was: (i) developed, (ii) modeled and (iii) calibrated. Anthrob replicates the human upper limb and features four skeletal DoF (one revolute elbow and one spherical glenohumeral joint). Furthermore, it is equipped with 11 muscles¹ that are implemented by electromagnetic series-elastic actuators. While 10 out of the 11 muscles are uni-articular (i. e. they only affect one joint), Anthrob also features one bi-articular muscle which spans and actuates the glenohumeral and elbow joint simultaneously. Anthrob, which was conceptualized at the end of the Eccerobot project, was developed together with Michael Jäntsch and the Swiss company awtec AG.

To evaluate the physics-based simulation approach, three Anthrob physics-based models were developed. All models used an identical skeleton model but featured different muscle kinematic models. While model I relied exclusively on straight-line muscle kinematics, model II and model III extended model I by cylindrical and mesh wrapping surfaces, respectively. Even though most model parameters could be deduced from the available data sheets and the CAD model, three additional identification experiments were required and conducted to derive models of: (i) the stress/strain properties of the NBR O-rings used as Series Elastic Elements (SEEs), (ii) the friction properties of the ceramic eyelets used to guide the Dyneema kite-line and (iii) the input/output properties of the power circuit of the Electronic Control Units (ECUs).

The simulation-reality gap of the three derived Anthrob models was finally analyzed and minimized using the presented calibration procedure. The gap was quantified by measuring the joint position and muscle force errors of the models for both, static robot postures and dynamic robot trajectories. To measure the joint position of the spherical glenohumeral joint of the Anthrob robot, an extrinsic stereo-vision, infrared-marker based motion capture system with realtime capabilities was developed. In the uncalibrated state, the straight-line muscle kinematics model was out-performed by both wrapping surface models with even lower errors for the mesh wrapping surfaces. After calibration, the differences between the three models became smaller when steady robot postures are considered. In the dynamics case, the wrapping surface models again out-performed the straight-line model significantly, eventually leading to a final average joint position error in the order of $\sim 5^\circ$ and $\sim 2^\circ$ for the spherical glenohumeral and the revolute elbow joint, respectively. These errors, which are large by means of industrial robots, are low for the class of musculoskeletal robots where a repeatability of 3-5° is common. Hence, it can be concluded that the physics-based simulation approach is a

The musculoskeletal robot Anthrob

Anthrob modeling

Anthrob model simulation-reality gap & calibration

^{1 13} when the hand is attached

140 CONCLUSIONS

promising substitute for classical joint-space approaches, which facilitates the investigation of aspects of the musculoskeletal design in a well-defined environment and finally will hopefully contribute to the promotion of this class of robots to a greater audience.

9.2 FUTURE WORKS

Evaluation of physics-based approach on more complex robot Even though the physics-based simulation approach was successfully applied to the Anthrob robot, the question remains how the approach performs if more complex robots are considered. Here, the main limiting factor is the number of skeletal DoF as additional muscles only increase the computational load. However, preliminary simulations of the Ecce-2 with its 103 DoF conducted by project partners of the Eccerobot consortium showed promising results—even though no simulation-reality gap analysis has been performed so far.

Realtime performance of muscle models Another aspect, that has to be considered if more complex robots are targeted, is the execution time of the muscle kinematics models. In Chapter 7 it was shown that a single simulation step of the developed Anthrob skeleton model required 18 µs whereas the simulation of a single straight-line muscle and a muscle with a mesh wrapping surface within the muscle path varied between 4 µs and 3 ms, respectively. Hence, depending on the used muscle kinematics model, the complete Anthrob robot with 11 muscles can be simulated in realtime if the simulation frequency is lower than 30 Hz for the mesh wrapping surface kinematics and lower than 16 kHz for straight-line muscle kinematics. However, particularly the mesh wrapping surface frequency is a worst-case estimate. The execution time of this muscle kinematics model is highly dependent on the mesh density and the robot posture which in turn determines the number of mesh vertices that have to be considered for the path computation. In reality, the complete Anthrob mesh wrapping surface model executed with frequencies between 80 Hz and 120 Hz. The simulation frequency is important as the simple numerical integrators used in physics-engines induce simulation artifacts if the simulation frequencies are too low and this lower frequency bound depends on the complexity of the model. In case of the Anthrob robot, simulation frequencies below 500 Hz are not advisable. Thus, realtime simulations of the mesh wrapping muscle kinematics model are currently not possible without compromising the accuracy. However, the computation of the muscle dynamics can be perfectly parallelized on multiple Central Processing Units (CPUs). With this technique and faster CPUs it should be soon possible to run models of similar complexity than the Anthrob model in realtime-also when mesh wrapping surfaces are used.

Jäntsch et al. already successfully employed the developed models as a software-in-the-loop tool for the development and testing of novel control strategies [60]. But other use-cases for physics-based simulations of musculoskeletal robots are also conceivable. For instance, the simulation model could be used as an internal model for control to predict possible interactions with the robot environment or to derive motor commands required for a desired movement. This approach, which is reminiscent of the internal model control hypothesis known from Neuroscience [68], is particularly elegant to implement on a physics-engine as the required tools for the simulation of the environment, namely rigid body dynamics, are already provided outof-the-box. Physics-based model as internal model

REAL-TIME MOTION CAPTURE

In Section 6.3 it was pointed out that no commercial, intrinsic sensors are available to measure the joint positions of spherical joints. Therefore, an extrinsic substitute using a stereo-vision system was developed based on [36] and [37]. The system uses passive retro-reflective markers that are mounted to the links of the robot and that are illuminated by infrared light: (i) to track the links and (ii) to compute the joint position of the spherical glenohumeral joint of the Anthrob robot.

Existing motion capture systems can be classified in either *inside-out* or outside-in trackers [118]. Whereas in inside-out trackers the camera is mounted to the moving artifact (e.g. the head of a person) and fix markers in the environment are tracked, in outside-in trackers the opposite is the case. For our application, where the work environment is well-defined and constant, an outside-in tracker should be favored as mounting cameras to the robot links would have a significant impact on the inertia of the links and therefore on the robot dynamics. Similar outside-in trackers have been developed and reported by Dorfmüller [25], Ribo et al. [127] and Pintaric and Kaufmann [118]. However, all systems have been developed for pose tracking in virtual or augmented reality scenarios and do not provide joint position data. Commercial systems are also available on the market, such as the optical tracker available from Vicon or the magnetic tracker developed by Polhemus. But, both systems are quite expensive and the magnetic tracker approach was also not suitable for our system where the metal parts of the robots and the DC motors interfere with the magnetic fields of the tracker [36].

A.1 HARDWARE SETUP

The hardware setup of the motion capture system is shown in Figure A.1. It comprises two monochrome PointGrey Flea-3 cameras (FL3-FW-03S1M-C) with a resolution of 648x488 pixel (0.3 megapixel), a global shutter and a maximum frame rate of 120 FPS. The cameras are equipped with Schneider FIL093 infrared pass filters that have a cut-on wavelength (the wavelength at which the transmission is 50%

Related works

Cameras





Figure A.1: Motion capture hardware setup. The developed motion capture system consists of two PointGrey Flea3 cameras that are equipped with infrared pass-filters and Pentax H612A optics. Both cameras are enclosed by infrared spot lights and are mounted to a Bosch profile via custom-built carriers.

of the maximum) of 830 nm. As optics, Pentax H612A lenses with an aperture of F1.2 and focal length of 6 mm are used. Furthermore, each camera is enclosed by a cluster of six Kingbright infrared spot lights (Kingbright BL0106-15-28) with a peak wavelength of 940 nm to illuminate the retro-reflective marker targets mounted to the robot.

- Stereo rig The two Flea-3 cameras are mounted to a Bosch profile via custombuilt carriers that facilitate the adjustment of both the stereo rig baseline (the distance between the camera centres) and the camera orientation (in the plane of the Bosch profile surface, see Figure A.1). The two cameras are connected to a single Firewire 800 (IEEE 1394b-2002) bus via a Peripheral Component Interconnect Express (PCIe) interface card which ensures an automatic trigger synchronization with a maximum jitter of 125 µs¹. This is crucial, as any time jitter between the two camera images would induce artifacts in the computed 3D measurements (see Section A.3.3), rendering the resulting joint angles erroneous.
- *Markers and marker targets* Passive, retro-reflective spherical markers with a diameter of 16 mm were used for the marker targets—i. e. the rigid constellation of $k \ge 3$ markers that are mounted to a single robot link via threaded rods. However, as the arrangement of the markers has a significant impact on the resulting tracker performance, particular emphasis was put on selecting the marker positions. Aspects that were considered are the marker visibility (ideally, all markers should be visible in both camera images at all times) and the uniqueness of the mutual

¹ see Flea3 datasheet available on http://www.ptgrey.com for an explanation

Euclidean marker distances (required for marker labeling—see Section A.3.4). Even though theoretically three markers per marker target are sufficient for pose estimation if the marker arrangement is nonsymmetrical [135], five markers per target were used for the Anthrob robot to increase the accuracy and robustness of the motion capture system.

A.2 CALIBRATION

In this section the methods used for the calibration of the internal and external parameters of the stereo rig as well as of the marker targets are summarized.

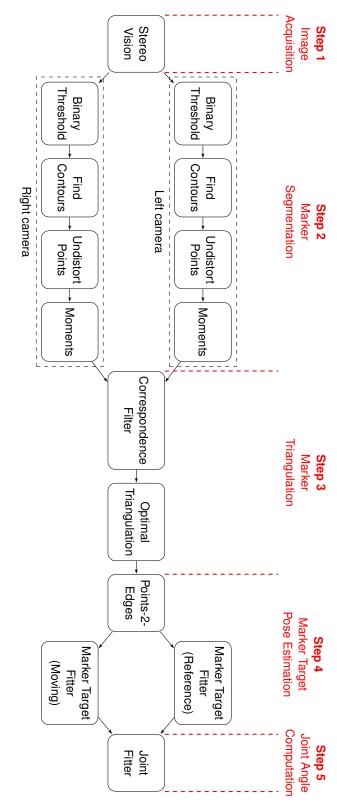
The calibration of the stereo rig is performed in two steps. First, the internal calibration matrices as well as the radial and tangential lens distortion coefficients of each camera are estimated using a set of checkerboard calibration images and the OpenCV function calibrateCamera (see [176] and [9] for details about the calibration algorithm). In the next step, the required parameters of the stereo rig—namely the essential and fundamental matrix (*E* and *F*, respectively) as well as the transformation matrix ${}^{L}T_{R}$ between the right and left camera—are estimated using a third set of checkerboard images and the OpenCV function stereoCalibrate.

As described in Section A.1, each robot link is equipped with k spherical, retro-reflective markers that constitute a marker target. In order to identify these marker targets during tracking, the k(k - 1)/2 mutual Euclidean distances between the markers are used (see Section A.3.4). Therefore, the Euclidean distances have to be identified a priori for each marker target. This is done by manually labeling the markers of a single target in a set of calibration images and subsequent optimization of the spatial distances between the modeled and computed 3D coordinates of the marker centroids, as described in detail in [36].

A.3 IMAGE PROCESSING PIPELINE

The image processing pipeline of the motion capture system is shown in Figure A.2. It consists of stages (e.g. a binary threshold transformation) which can be grouped into the following five processing steps: (i) image acquisition, (ii) marker segmentation, (iii) marker triangulation, (iv) marker target pose estimation and (v) joint angle computation. In the image acquisition step, synchronized monochrome images are acquired which are then processed by the marker segmentation step to determine the 2D coordinates of the marker centroids. In the subsequent marker triangulation step, these 2D point sets (from the left and right camera, respectively) are filtered to find the corresponding 2D point pairs and the 3D coordinates of each pair *Calibration of the stereo rig*

Calibration of the marker targets



a binary threshold transformation) which can be grouped into five processing steps: (i) image acquisition, (ii) marker segmentation, (iii) marker triangulation, (iv) marker Figure A.2: Image processing pipeline of the developed motion capture system. The image processing pipeline of the motion capture system consists of stages (e.g. target pose estimation and (v) joint angle computation. Each pipeline stage is implemented as a QptStage for maximum performance (see Appendix F).

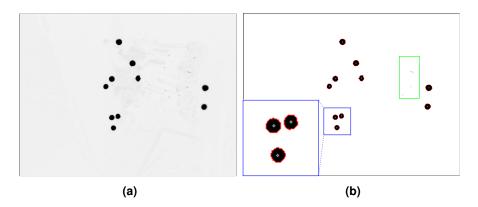


Figure A.3: Marker segmentation. (a) Raw monochrome camera image (colors are inverted). The spherical retro-reflective markers appear as dark spots in the image. (b) Image after applying the marker segmentation algorithm. First, the monochrome image from (a) is converted to a binary image by applying a binary threshold to the pixel brightness. Next, the marker contours are detected (red lines) and finally the 2D marker centroids (white dots) are computed via the spatial moments of the contour points. To improve the performance of the system, false-positive contours, due to infrared reflections, are detected and dropped (e.g. green area).

are computed via optimal triangulation. In the following marker target pose estimation step, the unlabeled 3D coordinates are processed to determine the 3D points that constitute a marker target and its current pose. Finally, the joint angles are computed from the marker target poses. All five processing steps are described in detail in the following sections.

A.3.1 Image Acquisition

Synchronized pairs of camera images are acquired from the stereo rig using the open-source library libdc1394, which provides a fast, high-level Application Programming Interface (API) for interfacing IEEE1394 cameras. To minimize the latency of the image acquisition step, the shutter time of the cameras was set to 0.03 ms (the shortest exposure time possible). This was possible due to the bright illumination of the retroreflective marker balls by the two clusters of infrared spot lights (see Figure A.1).

A.3.2 Marker Segmentation

In computer vision, "image segmentation is the task of finding groups of pixels that go together" [150]. Accordingly, marker segmentation aims at detecting the individual markers in the two stereo images and computing the 2D coordinates of their centroids. Fortunately, due to the infrared illumination and the resulting high contrast between the markers and the background (see Figure A.3a), marker pixels can be identified in the monochrome images by a binary threshold ap-

plied to the pixel brightness-similar to other infrared-based tracking systems [118, 127]. The resulting binary images are subsequently processed by the "border following" algorithm of Suzuki et al. [149], implemented by the OpenCV function findContours, to extract the contours of the image regions that possibly constitute a marker. "Possibly", as there might be false-positive contours due to infrared light reflections from other objects (e.g. see green rectangle in Figure A.3b). These false-positives are filtered partially by constraining the number of contour pixels per detected region. This is possible as the range distance of the marker targets with respect to the cameras and therefore the size range of the marker blobs in the images can be determined a priori. Remaining false-positive contours will be handled during the marker labeling in Section A.3.4. In the next pipeline stage, the contour points are corrected for radial and tangential lens distortion to obtain their ideal image positions (which obey linear projection) by applying the OpenCV function undistortPoints. Thereafter, the centroids $c \in \mathbb{R}^2$ of the detected contours are computed via the spatial moments of the contours using the OpenCV function moments (see also [172]).

A.3.3 Marker Triangulation

Correspondence filtering In order to compute the 3D coordinates of the markers from the identified and undistorted marker centroids, the corresponding centroids in each stereo image pair have first to be found. This is done by applying the epipolar constraint to all centroid pairs, similar to [25, 118, 127]. Consider a pair of undistorted marker centroids from the left and right camera image (c and c', respectively) and the fundamental matrix F. Then the epipolar constraint is given by:

$$c'^T F c = 0 \tag{A.1}$$

Geometrically, this means that the marker centroid c', projected on the image plane of the second camera, lies on the epipolar line defined by *Fc*. However, this approach suffers from two problems. First, the epipolar constraint is only a necessary but not a sufficient condition, i. e. there might be multiple 2D coordinates that fulfill the constraint. Secondly, the centroid coordinates (*c* and *c'* in our example), are subject to noise and therefore the equality condition of the epipolar constraint might never by fulfilled—even for the correct centroid pair. While the former problem will be tackled in the marker target pose estimation step (see Section A.3.4), the latter is addressed by introducing a threshold δ for the maximum distance to the epipolar line, which yields:

$$c'^T F c \le \delta \tag{A.2}$$

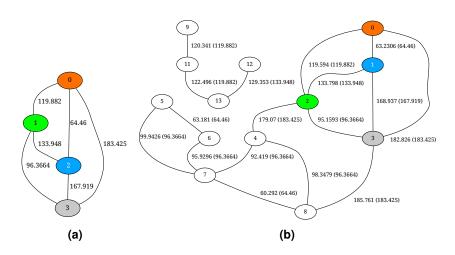


Figure A.4: Example of the developed graph-based marker labeling procedure for a marker target with four retro-reflective markers. For each marker target a complete undirected graph, called the reference graph, is determined before runtime by using the 3D positions of the marker centroids as nodes and their mutual Euclidean distances as edge weights as shown in (a). Then, the marker labeling is performed by a maximum clique search of the reference graph in the candidate graph shown in (b), which is created for each stereo image pair at runtime.

Clearly, high values for δ result in false-positive correspondences which slow down the remaining processing steps. On the other hand, a low value for δ might lead to false-negatives and, in the worst case, to no matches at all. Therefore, a good balance has to be found to ensure a low pipeline latency while retaining a high correspondence rate. Here, based on empirical investigations, a maximum tolerated distance of $\delta = 1$ pixel has been chosen.

Once the correspondence candidates for each marker centroid are obtained, the 3D coordinates of these correspondences can be computed. Here, the optimal triangulation algorithm developed by Hartley and Sturm [42] and implemented by the OpenCV functions cvCorrect-Matches and cvTriangulatePoints has been used to obtain the set *C* of marker centroids $\tilde{c}_i \in \mathbb{R}^3$ for the current image pair:

$$C := \{ \tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_n \} \qquad n \in \mathbb{N}$$
(A.3)

A.3.4 Marker Target Pose Estimation

In order to estimate the current pose of the marker targets, the markers have first to be labeled (i. e. the 3D points that belong to a marker target have to be determined). Once these 3D points are selected from the set of computed 3D points, the current pose of the marker target can be estimated.

Each robot link is equipped with k spherical markers that constitute a marker target. Marker labeling is the task of identifying the k markers of a marker target from the set of computed marker centroids C. In

Optimal triangulation

Marker labeling

the literature, different approaches to this problem can be found (see for instance [25], [127] or [118]). Here, a novel, graph-based approach that aims at identifying the marker targets in a complete, undirected and weighted graph generated from the set *C* of 3D points is presented. From a mathematical point of view, this problem is very similar to the clique problem in computer science which was proven to be NP-complete by Richard Karp in 1972 [67].

In detail, the algorithm proceeds as follows. First, each calibrated marker target (see Section A.2) is converted into an undirected complete graph, called the reference graph G_{ref} , by using the k markers as vertices and the mutual Euclidean distances between the markers as edge weights. This has to be done only once per marker target before runtime. An example of such a marker target reference graph is shown in Figure A.4a. Then, the marker centroids \tilde{c}_i , which are possible candidates for the vertices of the reference graph G_{ref} , are identified. Here, the search space is first pruned by calculating the n(n-1)/2 Euclidean distances between the *n* marker centroids of *C* and selecting only the *m* centroids \tilde{c}_i out of *C* that have edge correspondences in the reference graph G_{ref} . By edge correspondence it is meant that the Euclidean distance between two marker centroids $(\tilde{c}_a \text{ and } \tilde{c}_b, \text{ respectively})$ is within a threshold ϵ of one of the edge weights of the reference graph G_{ref} . This way a subset $D \subseteq C$ is obtained that contains only the centroids that are possible candidates for the vertices of G_{ref} . It should be noted, that the computation of Euclidean distances between the marker centroids \tilde{c}_i is required only once per stereo image pair, independent of the number of tracked marker targets. Therefore, this computation is performed in a dedicated pipeline stage, called Points-2-Edges, before the actual marker labeling (see Figure A.2). As threshold for the edge length $\epsilon = 3 \text{ mm}$ is used which substantially reduces the search space without affecting the marker labeling. In the next step, the m centroids of D are converted into an undirected, weighted graph, called the candidate graph $G_{\rm c}$, using again the centroids as vertices and the mutual Euclidean distances between two centroids as edge weights (similar to the reference graph G_{ref}). However, due to the pruning of the search space, G_c is not a complete graph anymore. An example of such a candidate graph generated from a pair of stereo images is shown in Figure A.4b. Once both of these graphs are constructed, the markers are labeled by finding the largest subgraph of the reference graph $G_{\rm ref}$ in the candidate graph $G_{\rm c}$. Here a recursive implementation of a depth-first search algorithm is used to detect all *l* cycles in the candidate graph $G_{\rm c}$ under the constraint that the edge weights of the cycle match the corresponding edge weights of the reference graph G_{ref} .

Once all *l* cycles are detected, the transformations T_i (i = 1,...,l) between the labeled markers and the 3D positions of the markers obtained during marker target calibration (see Section A.2) are estimated for each cycle using the algorithm of Umeyama [156]. Finally, the "best" cycle is determined by computing the average Euclidean distance between the detected markers and the reference positions determined during marker target calibration and selecting the cycle with the lowest error.

A.3.5 Joint Angle Computation

In the final step of the image processing pipeline, the joint angles are computed from the estimated poses of the marker targets.

If we are only interested in the relative orientation of the moving marker target with respect to the reference marker target (e.g. the humerus with respect to the scapula), the computation of the joint position of a spherical joint is trivial. Consider the rotation matrices of the reference and moving marker targets of a joint are given in the camera frame ($^{cam}R_{ref}$ and $^{cam}R_{mov}$, respectively). Then the orientation of the moving marker target with respect to the reference marker target $^{ref}R_{mov}$ is equal to:

$${}^{\text{ref}}R_{\text{mov}} = {}^{\text{ref}}R_{\text{cam}} {}^{\text{cam}}R_{\text{mov}} = {}^{\text{cam}}R_{\text{ref}}^{-1} {}^{\text{cam}}R_{\text{mov}}$$
(A.4)

However, this approach is only viable if no error estimate for the offset between the ball and the socket of the joint is required. Otherwise, alternative approaches, as presented in [36], should be considered.

A.4 PERFORMANCE BENCHMARKING

In this section the performance of the developed motion capture system will be assessed and compared to the reference system developed by Gaschler [36, 37].

To maximize the performance of the motion capture system, a sharedmemory based pipeline toolkit, called Qt Pipeline Toolkit (QPT), was developed and used for the implementation of the motion capture system. QPT is based on Qt and provides classes for the efficient implementation of pipeline stages (called QptStage) and inter-stage communication via templated data in- and outports (for details, see Appendix F). Therefore, each processing stage of the motion capture pipeline (see Figure A.2) has been implemented as a dedicated QptStage.

The image processing latency t_L of the developed motion capture system—the time required for the processing of a single stereo image pair—as well as of the individual processing steps was analyzed

Pose estimation

Computation of spherical joint positions

Implementation using QPT

Image processing latency

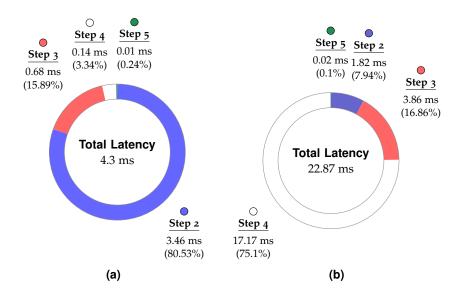


Figure A.5: Image processing pipeline latency. Total latency and latency of the individual processing steps for the developed motion capture system (**a**) and the reference system developed by Gaschler [36, 37] (**b**). For the latency analysis, a sequence of 590 stereo images was recorded a-priori and fed-back to both systems. The developed system out-performs the reference system by more than 500 %.

and compared to the reference system developed by Gaschler [36, 37]. Therefore, an image sequence of 590 stereo pairs was recorded and fed back into both algorithms. The benchmarks were performed on a desktop computer equipped with an Intel[®] Core^{TM} i7-860 processor. The results of the latency analysis are shown in Figure A.5 (step 1 of the pipeline—the image acquisition step—was not included in the analysis due to the use of pre-recorded images). It can be seen that, even though both systems use a similar approach (stereo rig and passive, retro-reflective markers), a t_L decrease of $\approx 532\%$ (4.3 ms vs. 22.87 ms for the reference system) was achieved. Even though this can be partially accounted to the efficient implementation of the system using QPT, the main factor responsible for the decrease is the developed, graph-based marker target pose estimation procedure (pipeline step 4, see Section A.3.4). While the reference system required 17.17 ms (75.1% of t_L), the graph-based approach only took 0.14 ms (or 3.34 % of t_L) for the same task. However, also two of the remaining three pipeline steps (3 and 5, respectively) are faster than in the reference system. In fact, the only processing step where the developed system underperforms in comparison to the reference system is the marker segmentation step (step 2, 3.46 ms vs. 1.82 ms). The reason for this is that in [36], the thresholding and marker region detection is performed in a single step. This, however, is not possible in the presented stage-based pipeline approach that relies on standard functions provided by OpenCV.

Accuracy of marker In addition to the pipeline latency, the accuracy of the marker segmensegmentation tation was analyzed. Again, an image sequence of 590 stereo images

Table A.1: Accuracy of marker segmentation. The accuracy of the marker segmentation pipeline step was assessed and compared to the reference system developed by Gaschler [36, 37] by computing the mean pixel coordinates and standard deviations of the 2D marker centroids for a sequence of 590 images (recorded for a steady robot posture under constant lighting conditions).

Marker		Wittme	eier		(Gaschler [3	6, 37]	
	\bar{c}_x	\bar{c}_y	σ_{x}	σ_y	\bar{c}_x	\bar{c}_y	σ_x	σ_y
1	198.549	333.929	0.015	0.013	198.546	333.955	0.005	0.005
2	203.979	302.628	0.027	0.028	203.974	302.668	0.005	0.004
3	233.171	298.712	0.022	0.028	233.168	298.752	0.005	0.005

was recorded and fed back to the motion capture algorithms. However, in contrast to the image sequence used for the latency analysis, both the lighting conditions and the robot posture was constant for all images. The results of this test for the left camera images and three markers are shown in Table A.1. It can be seen that all markers are detected by both systems and that the marker centroids averaged over the 590 images (\bar{c}_x and \bar{c}_y , respectively) are almost identical (coordinate deviations are at the order of 1×10^{-3} pixel). However, the table also shows that the marker centroid computation algorithm used by Gaschler [36, 37] results in a lower standard deviation for the marker centroid (σ_x and σ_y , respectively). The reason is that Gaschler used a linear threshold instead of a binary threshold for the marker segmentation which results in smoother contours at the marker edge and therefore a more stable centroid computation. However, also the binary threshold used for the presented motion capture system results in extremely low and therefore sufficient standard deviations in the order of 1×10^{-2} pixels.

A.5 SUMMARY

A marker-based motion capture system with real-time capabilities was developed. Hence, particular emphasis was put on the accuracy, robustness and latency of the used image processing routines and a total speed-up of more than 500 % was achieved—compared to the reference system developed by Gaschler [36, 37]. Furthermore, it was shown that the main reason for the speed-up was the developed, graph-based marker labeling algorithm which out-performs the reference system by two orders of magnitude. Even though the motion capture system has been developed as an extrinsic joint position sensor for the glenohumeral joint of the Anthrob robot, the low latency and high robustness of the system makes it also perfectly suitable for other motion capture usage scenarios, such as virtual or augmented reality applications.

LINEAR VISCOELASTICITY

B.1 STRESS AND STRAIN

Consider a specimen that is put under a tensile load f perpendicular to the cross section area of the specimen as shown in Figure B.1. The load f induces a *stretch* of the specimen, which is called the *strain* ε . Furthermore, the load is opposed by the *stress* σ , which is the reaction force of the internal bonds of the material per unit area. Two types of stresses and strains are commonly defined: (i) the engineering (or nominal) stress/strain and (ii) the true stress/strain. Both are introduced in the following paragraphs.

The engineering (or nominal) stress σ_e is defined as the force per unit area A_0 of the undeformed specimen (see Figure B.1a) [10, 86, 139]:

$$\sigma_e = \frac{f}{A_0} \tag{B.1}$$

Similar to [86], we use the convention that tensile stresses are positive and compressive stresses are negative. Stresses are expressed in units N/m^2 or more commonly in Pa.

Likewise, the engineering (or nominal) strain ε_e in the direction of the tensile load is defined as the integral of the increase in length dl of the specimen with respect to its initial length l_0 [10, 86, 139]:

$$\varepsilon_e = \int_{l_0}^{l_1} \frac{\mathrm{d}l}{l_0} = \frac{l_1 - l_0}{l_0} = \frac{l_1}{l_0} - 1 \tag{B.2}$$

If a material exhibits large deformations, such as rubbers or biological tissues, the strain is sometimes expressed in terms of the expansion ratio λ [86, 139], equal to:

$$\lambda = \varepsilon_e + 1 \tag{B.3}$$

In contrast to the engineering stress and strain, the true stress σ_t and strain ε_t are calculated with respect to the current cross-section area A_1 and length l of the deformed specimen (see Figure B.1b). Therefore, the true stress σ_t is defined as [10, 86]:

$$\sigma_t = \frac{J}{A_1} \tag{B.4}$$

Engineering stress and strain

True stress and

strain

B

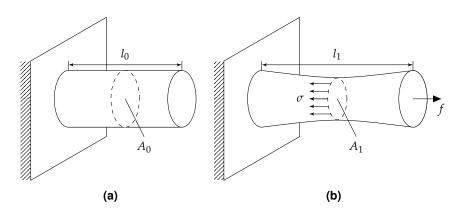


Figure B.1: Stress and strain. (a) Cylindrical specimen under rest conditions. (b) Specimen under tensile load *f*. The induced stretch of the specimen is called the *strain* ε , whereas the load opposing reaction of the material is the *stress* σ .

and the true strain ε_t is equal to [10, 86]:

$$\varepsilon_t = \int_{l_0}^{l_1} \frac{\mathrm{d}l}{l} = \ln \frac{l_1}{l_0} = \ln \left(1 + \varepsilon_e\right) \tag{B.5}$$

B.2 BASIC MODELS

Basic mechanical elements In linear viscoelastic theory, two basic mechanical elements form the constituents of phenomenological models [10, 86, 139]: (i) a Hookean spring that is used to model the elastic portion of a viscoelastic material and (ii) a Newtonian damper (or dashpot) which represents its viscous or fluid mechanical properties. In a Hookean spring the stress σ_s is linearly proportional to the strain ε_s with the slope defined by Young's modulus *E*:

$$\sigma_s = E \,\varepsilon_s \tag{B.6}$$

In contrast, the stress of a dashpot σ_d does not linearly depend on the strain but on the strain rate $d\varepsilon_d/dt$ with the slope defined by the viscosity μ :

$$\sigma_d = \mu \, \frac{\mathrm{d}\varepsilon_d}{\mathrm{d}t} \tag{B.7}$$

Maxwell fluid model The Maxwell fluid model is a serial combination of a linear spring and dashpot as shown in Figure B.2a. Here, the applied stress σ_M under equilibrium conditions is equal to the stress in the spring σ_s and the dashpot σ_d , respectively:

$$\sigma_M = \sigma_s = \sigma_d \tag{B.8}$$

Furthermore, the strain of a Maxwell fluid ε_M is equal to the sum of the spring ε_s and dashpot strain ε_d :

$$\varepsilon_M = \varepsilon_s + \varepsilon_d \tag{B.9}$$

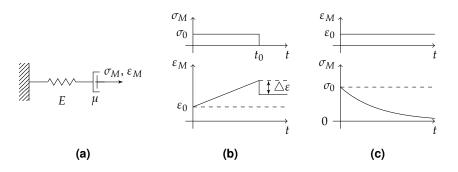


Figure B.2: Maxwell fluid model. (a) The Maxwell fluid model serially combines a Hookean spring with a Newtonian damper (or dashpot). (b) Creep and creep recovery response. The strain of the Maxwell model linearly increases under constant stress due to the continously expanding dashpot. If the applied stress is removed, the linear spring returns to its resting length while the dashpot expansion persists. (c) Relaxation response. Under constant strain, the Maxwell model shows an exponentially decreasing stress due to the expanding dashpot.

By differentiating Equation B.9 with respect to time, substituting the resulting strain rates with Equation B.7 and the time derivative of Equation B.6 and solving for σ_M yields the differential equation of the Maxwell fluid model [10, 18, 81, 139]:

$$\dot{\sigma}_M + \frac{1}{\tau} \sigma_M = E \,\dot{\varepsilon}_M \tag{B.10}$$

with $\tau = \mu/E$ known as the *relaxation time*. The behavior of viscoelastic models is typically analyzed by their response to creep and relaxation tests. While in the creep test the stress is kept constant and the strain is measured over time, the relaxation test is of greater importance in the context of this work. Here, the strain is kept constant and the resulting stress is measured over time. A qualitative response of this model to these creep and relaxation tests is shown in Figure B.2b and B.2c, respectively. Under constant stress σ_0 , the Maxwell fluid model shows a linearly increasing strain due to the continuous dashpot expansion. If the stress is immediately removed (creep recovery), the Maxwell fluid shows a negative strain step of magnitude $\Delta \varepsilon = \sigma_0/E$ as the Hookean spring returns to its resting position. However, as the dashpot expansion persists, the total strain ε_M remains positive.

The Voigt solid model, sometimes also referred to as the Kelvin solid model [10, 18, 81], combines a linear spring and a dashpot in parallel (see Figure B.3a). Hence, the equations for the overall stress σ_V and strain ε_V are given by:

 $\sigma_V = \sigma_s + \sigma_d \tag{B.11}$

$$\varepsilon_V = \varepsilon_s = \varepsilon_d \tag{B.12}$$

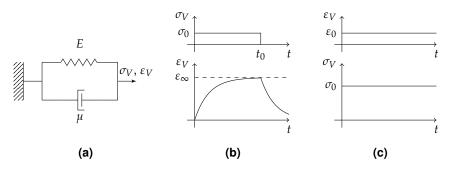


Figure B.3: Voigt (or Kelvin) solid model. (a) The Voigt solid model parallely combines a Hookean spring with a Newtonian damper (or dashpot). (b) Creep and creep recovery response. The strain of the Voigt model exponentially increases under constant stress due to the continously expanding dashpot. If the applied stress is removed, the strain exponentially declines as the spring and the dashpot return to their resting position. (c) Relaxation response. Under constant strain, the Voigt model shows a constant stress equal to $\sigma_0 = E \varepsilon_0$.

Again, replacing σ_s and σ_d in Equation B.11 with the basic constitutive equations of the spring and dashpot and subsequently solving for ε_V yields the standard differential equation of the Voigt model equal to:

$$\dot{\varepsilon}_V + \frac{1}{\tau}\varepsilon_V = \frac{\sigma_V}{\mu} \tag{B.13}$$

Example responses of this model to creep, creep recovery and relaxation experiments are shown in Figure B.3b and B.3c, respectively. In the creep test, the Voigt solid model shows an exponential increase in strain. This is different to the Maxwell fluid model which exhibited a discrete jump in the strain. In general, every model that contains a free spring will show a direct strain response to changes in the input stress similar to the Maxwell fluid model, while models with a spring in parallel to a dashpot will exponentially converge to the equilibrium strain ε_{∞} . Hence, in the context of the creep test, the constant τ is called the *retardation time* and is defined as the time required for the strain to come within 1/e of its asymptotic value [10]. When the applied stress is removed (creep recovery), the Voigt model exhibits an exponential decay as the spring is slowly returning to its resting length. In the relaxation test, the Voigt model shows a constant stress as Equation B.13 simplifies to $\sigma_0 = E \varepsilon_0$.

ANTHROB MODELS

C.1 SKELETON MODEL

<?xml version="1.0" encoding="UTF-8"?> <COLLADA xmlns="http://www.collada.org/2005/11/COLLADASchema" version="1.4.1"> <node id="humerus" name="humerus"> <matrix> -0.99621 0 0.08528 0 0.08528 0 0.99621 -0.218 0 1 0 0.19 0 0 0 1 </matrix> </node> <node id="base" name="base"> <matrix> 0.707 -0.707 0 -0.154 0.707 0.707 0 -0.064 0010 0001 </matrix> </node> <node id="forearm" name="forearm"> <matrix> -0.99621 0 0.08528 0 0.08528 0 0.99621 -0.218 0 1 0 0.19 </matrix> </node> </visual_scene> </library_visual_scenes> <library_physics_models> <technique_common> <dynamic>false</dynamic> <mass>0</mass> <inertia>0.014939171 0.190795642 0.199221684</inertia> </technique_common </rigid_body> <rigid_body name="Humerus-RigidBody" sid="Humerus-RigidBody"> <technique_common> <dynamic>true</dynamic>
<mass>0.70339</mass> <mass_frame> wass-11000ex <translate>0.00251 0.09431 -0.00031</translate> <rotate>-0.342953544733758 -0.347996979215136 0.872514165278530 97.462368068991239</rotate> </mass_frame> <inertia>0.000645324 0.00226816 0.002759107</inertia> </technique_common> </rigid_body> <rigid_body name="Forearm-RigidBody" sid="Forearm-RigidBody"> <technique_common> <dynamic>true</dynamic> <mass>0.11106</mass> <mass_frame> <translate>0.00099 -0.0778 -0.00163</translate> <rotate>-0.029689875993894 0 0.999559158461102 90.595365177020312</rotate> </mass_frame> <inertia>0.000014439 0.000399272 0.000404443</inertia> </technique_common> </rigid_bodv> <rigid_constraint name="Shoulder" sid="Shoulder"> <ref_attachment rigid_body="Base-RigidBody"> <translate>0 -0.218 0.41</translate> <rotate>0 0 1 0</rotate>

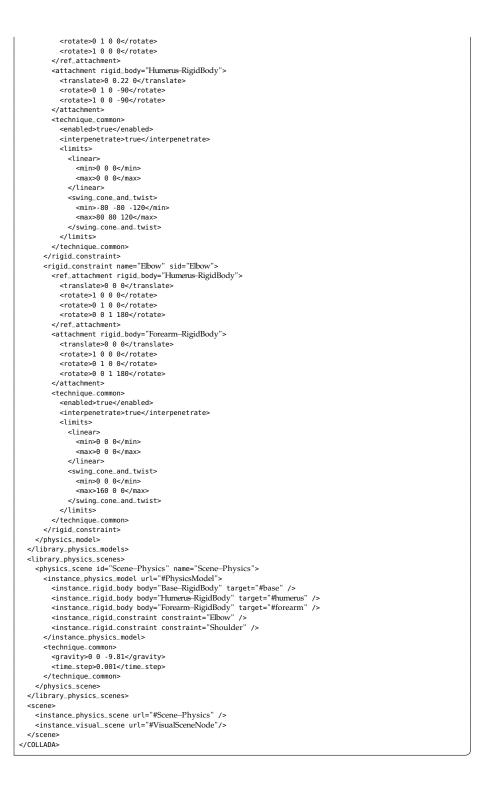


Table C.1: Joint friction parameters of the physics-based Anthrob model.

Joint		Stribeck Frict	ion Parameter	
	μ_s	μ_k	μ_v	v_s
Glenohumeral	0 N m	0 N m	0.15 Nms/rad	0.05 rad/s
Elbow	0 N m	0 N m	0.02 N m s/rad	0.05 rad/s

C.2 MUSCLE MODELS

		Tabl	e C.2: AC	ctuator typ	bes and p	aramete	rS.			
			DC moto	r		G	ear		Spindle	ECU
Туре	R_A (Ω)	L _A (H)	K _E (Vs/rad)	K_T (Nm/A)	J_M	J_G (kg m ²)	η (1)	N (1)	r _S (mm)	$R_{\rm ECU}$ (Ω)
A		5.73 · 10 ⁻⁵	. ,	. ,			. ,	. ,		
В	4.69	$0.288 \cdot 10^{-3}$						128		0.12

Table C.2: Actuator types and parameters.

Table C.3: Parameters of the Anthrob PD actuator position controller.

Туре	Muscles	Control Parameters
1	Lateral Deltoid, Anterior Deltoid, Posterior Deltoid, Supraspinatus, Infraspinatus, Teres Major, Teres Minor, Pectoralis, Biceps	
2	Triceps, Brachialis	$ v_{max} = \pm 12 \text{ V}, K_P = 0.025 \text{ V}/^{\circ}, K_D = 0.01 \text{ V} \text{ s}/^{\circ}, K_f = 40.21 \text{ mV/N} $

Table C.4: NBR-70 Zener model fitting results.

$l_0 \times d \text{ (mm)}$	10x4	10x5	12x4	12x5	15x4	15x5
E_0 (MPa)	μ 1.0720	0.6940	0.9778	1.0320	0.8968	0.8129
	σ 0.0201	0.0117	0.0482	0.0692	0.0381	0.0531
E_1 (MPa)	μ 0.2422	0.2116	0.2002	0.2033	0.1683	0.1375
	σ 0.0165	0.0144	0.0163	0.0173	0.0100	0.0123
$\tau_1(s)$	μ 1.3845	1.3128	1.2177	1.4590	1.3202	1.2886
• • •	σ 0.1507	0.1493	0.1686	0.1498	0.1513	0.1487

Table C.5: NBR-70 Wiechert model fitting results.

$l_0 imes d \text{ (mm)}$		10x4	10x5	12x4	12x5	15x4	15x5
E ₀ (MPa)	μ	1.0566	0.6810	0.9654	1.0177	0.8862	0.8045
	σ	0.0189	0.0127	0.0485	0.0685	0.0386	0.0533
E ₁ (MPa)	μ	0.2218	0.1975	0.1931	0.1838	0.1541	0.1272
	σ	0.0317	0.0300	0.0337	0.0276	0.0198	0.0199
E ₂ (MPa)	μ	0.1527	0.1300	0.1169	0.1297	0.1019	0.0827
2 (σ	0.0084	0.0078	0.0086	0.0087	0.0058	0.0064
τ ₁ (s)	μ	0.2355	0.2300	0.2287	0.2515	0.2428	0.2366
1.07	σ_1	0.0577	0.0520	0.0598	0.0592	0.0415	0.0410
τ ₂ (s)	μ	3.0195	2.9292	2.9207	3.2223	3.0015	2.9414
2. (*)	σ	0.3368	0.3578	0.4799	0.3734	0.2481	0.3498

	_		Muscle Kinematics		Muscle K	Muscle Kinematics					Tendon	don		SEE		Actuator
Muscle	0	Origin	Via-J	Via-Point 1	Via-P	Via-Point 2	Via-Point 3	oint 3	Insertion	tion					_	
	В	<i>p</i> (m)	B	<i>p</i> (m)	В	<i>p</i> (m)	В	<i>p</i> (m)	в	<i>p</i> (m)	<i>l</i> _{T0} (m)	r _T (mm)	Туре	Quantity	Segment	Туре
Lateral Deltoid	Base	$\begin{bmatrix} -0.0060 \\ -0.0287 \\ 0.4656 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0060 \\ -0.0557 \\ 0.4643 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0000 \\ -0.2570 \\ 0.4500 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0090 \\ 0.1475 \\ -0.0090 \end{bmatrix}$	0.3295	0.5	15×5	2		A
Anterior Deltoid	Base	$\begin{bmatrix} 0.0643 \\ -0.0288 \\ 0.4084 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0626 \\ -0.0557 \\ 0.4088 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0430 \\ -0.1540 \\ 0.4308 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0090 \\ 0.1475 \\ -0.0090 \end{bmatrix}$	0.2430	0.5	15×5	2	1	A
Posterior Deltoid	Base	$\begin{bmatrix} -0.0532 \\ -0.0288 \\ 0.4398 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0520 \\ -0.0558 \\ 0.4389 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0351 \\ -0.2246 \\ 0.4270 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0090 \\ 0.1475 \\ -0.0090 \end{bmatrix}$	0.2780	0.5	15×5	2	1	A
Supraspinatus	Base	$\begin{bmatrix} -0.0657 \\ -0.0290 \\ 0.3963 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0638 \\ -0.0560 \\ 0.3967 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0485 \\ -0.1720 \\ 0.4189 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0098\\ 0.1922\\ -0.0171 \end{bmatrix}$	0.2190	0.3	12×5	1	1	A
Infraspinatus	Base	$\begin{bmatrix} -0.0611 \\ -0.0287 \\ 0.3525 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0595 \\ -0.0556 \\ 0.3548 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0528 \\ -0.1720 \\ 0.3837 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0106 \\ 0.1820 \\ -0.0202 \end{bmatrix}$	0.2161	0.3	12×5	1	1	А
Teres Minor	Base	$\begin{bmatrix} -0.0439 \\ -0.0297 \\ 0.3112 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0431 \\ -0.0564 \\ 0.3154 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0402 \\ -0.1710 \\ 0.3485 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0091 \\ 0.1702 \\ -0.0147 \end{bmatrix}$	0.2075	0.3	12×5	1	1	А
Teres Major	Base	$\begin{bmatrix} 0.0638 \\ -0.0279 \\ 0.3636 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0622 \\ -0.0548 \\ 0.3659 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0528 \\ -0.1540 \\ 0.3837 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0202\\ 0.1820\\ -0.0106 \end{bmatrix}$	0.2154	0.3	12×5	1	1	А
Pectoralis Major	Base	$\begin{bmatrix} 0.0486 \\ -0.0283 \\ 0.3214 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0477 \\ -0.0550 \\ 0.3256 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0402 \\ -0.1540 \\ 0.3485 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0108\\ 0.1520\\ 0.0108 \end{bmatrix}$	0.1870	0.3	12×5	1	1	A
Biceps Brachii	Base	$\begin{bmatrix} 0.0446 \\ -0.0288 \\ 0.4496 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0434 \\ -0.0558 \\ 0.4486 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0271 \\ -0.2052 \\ 0.4503 \end{bmatrix}$	Humerus	$\begin{bmatrix} -0.0170\\ 0.1390\\ 0.0170 \end{bmatrix}$	Forearm	$\left[\begin{matrix} 0.0000 \\ -0.0460 \\ 0.0050 \end{matrix} \right]$	0.4708	0.5	15×5	2	1	А
Triceps Brachii	Humerus	$\begin{bmatrix} -0.0097\\ 0.1360\\ -0.0345 \end{bmatrix}$	Humerus	$\begin{bmatrix} -0.0089\\ 0.1090\\ -0.0357 \end{bmatrix}$	Humerus	$\begin{bmatrix} 0.0000 \\ 0.0470 \\ -0.0300 \end{bmatrix}$			Forearm	$\begin{bmatrix} 0.0000 \\ -0.0008 \\ -0.0220 \end{bmatrix}$	0.1284	0.3	10 imes 4	1	2	в
Brachialis	Humerus	0.0097 0.1360 0.0345	Humerus	0.0089 0.1090 0.0357					Forearm	$\begin{bmatrix} 0.0000 \\ -0.0460 \\ 0.0050 \end{bmatrix}$	0.1703	0.3	15 imes 4	1	1	в

Table C.6: Parameters of muscle kinematics model I (the force sensor segment of all muscles is 0).

All position vectors are given relative to the visual node transforms of the COLLADA skeleton model (before the correction of the center of mass by the mass frame).

162

ANTHROB MODELS

J					Muscle Kinematics	inematics					Ten	Tendon		SEE		Actuator
Muscle	Ori	Origin	Via-I	Via-Point 1	Via-P.	Via-Point 2	Via-P	Via-Point 3	Inse	Insertion						_
	В	(m) q	В	(m) <i>d</i>	В	<i>p</i> (m)	В	(m) <i>d</i>	B	<i>p</i> (m)	l _{T0} (m)	r_T (mm)	Type	Quantity	Segment	Type
Lateral Deltoid	Base	$\begin{bmatrix} -0.0060 \\ -0.0287 \\ 0.4656 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0060 \\ -0.0557 \\ 0.4643 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0000 \\ -0.2570 \\ 0.4500 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0090 \\ 0.1475 \\ -0.0090 \end{bmatrix}$	0.3295	0.5	15×5	7	1	А
Anterior Deltoid	Base	$\begin{bmatrix} 0.0643 \\ -0.0288 \\ 0.4084 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0626 \\ -0.0557 \\ 0.4088 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0430 \\ -0.1540 \\ 0.4308 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0090\\ 0.1475\\ -0.0090 \end{bmatrix}$	0.2452	0.5	15×5	7	1	V
Posterior Deltoid	Base	$\begin{bmatrix} -0.0532 \\ -0.0288 \\ 0.4398 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0520 \\ -0.0558 \\ 0.4389 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0351 \\ -0.2246 \\ 0.4270 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0250 \\ -0.2350 \\ 0.4100 \end{bmatrix}$	Humerus	$\begin{bmatrix} -0.0090 \\ 0.1475 \\ -0.0090 \end{bmatrix}$	0.2809	0.5	15×5	2	1	V
Supraspinatus	Base	$\begin{bmatrix} -0.0657 \\ -0.0290 \\ 0.3963 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0638 \\ -0.0560 \\ 0.3967 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0485 \\ -0.1720 \\ 0.4189 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0280 \\ -0.2250 \\ 0.4000 \end{bmatrix}$	Humerus	$\begin{bmatrix} -0.0098\\ 0.1922\\ -0.0171 \end{bmatrix}$	0.2239	0.3	12×5	1	1	A
Infraspinatus	Base	$\begin{bmatrix} -0.0611 \\ -0.0287 \\ 0.3525 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0595 \\ -0.0556 \\ 0.3548 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0528 \\ -0.1720 \\ 0.3837 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0106\\ 0.1820\\ -0.0202 \end{bmatrix}$	0.2170	0.3	12×5	1	1	V
Teres Minor	Base	$\begin{bmatrix} -0.0439 \\ -0.0297 \\ 0.3112 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0431 \\ -0.0564 \\ 0.3154 \end{bmatrix}$	Base	$\begin{bmatrix} -0.0402\\ -0.1710\\ 0.3485 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0091 \\ 0.1702 \\ -0.0147 \end{bmatrix}$	0.2089	0.3	12×5	1	1	A
Teres Major	Base	$\begin{bmatrix} 0.0638 \\ -0.0279 \\ 0.3636 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0622 \\ -0.0548 \\ 0.3659 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0528 \\ -0.1540 \\ 0.3837 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0202 \\ 0.1820 \\ -0.0106 \end{bmatrix}$	0.2179	0.3	12×5	1	1	A
Pectoralis Major	Base	$\begin{bmatrix} 0.0486 \\ -0.0283 \\ 0.3214 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0477 \\ -0.0550 \\ 0.3256 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0402\\ -0.1540\\ 0.3485 \end{bmatrix}$			Humerus	$\begin{bmatrix} -0.0108\\ 0.1520\\ 0.0108 \end{bmatrix}$	0.1870	0.3	12×5	1	1	V
Biceps Brachii	Base	$\begin{bmatrix} 0.0446 \\ -0.0288 \\ 0.4496 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0434 \\ -0.0558 \\ 0.4486 \end{bmatrix}$	Base	$\begin{bmatrix} 0.0271 \\ -0.2052 \\ 0.4503 \end{bmatrix}$	Humerus	$\begin{bmatrix} -0.0170 \\ 0.1390 \\ 0.0170 \end{bmatrix}$	Forearm	$\left[\begin{array}{c} 0.0000\\ -0.0460\\ 0.0050\end{array}\right]$	0.4715	0.5	15×5	2	1	V
Triceps Brachii	Humerus	$\begin{bmatrix} -0.0097 \\ 0.1360 \\ -0.0345 \end{bmatrix}$	Humerus	$\begin{bmatrix} -0.0089 \\ 0.1090 \\ -0.0357 \end{bmatrix}$	Humerus	$\begin{bmatrix} 0.0000\\ 0.0470\\ -0.0300 \end{bmatrix}$			Forearm	$\begin{bmatrix} 0.0000 \\ -0.0008 \\ -0.0220 \end{bmatrix}$	0.1284	0.3	10 imes 4	1	2	B
Brachialis	Humerus	$\begin{bmatrix} 0.0097 \\ 0.1360 \\ 0.0345 \end{bmatrix}$	Humerus	0.0089 0.1090 0.0357					Forearm	$\begin{bmatrix} 0.0000 \\ -0.0460 \\ 0.0050 \end{bmatrix}$	0.1703	0.3	15×4	1	1	B

C.2 MUSCLE MODELS 163

Muscle	В	Between Previous	Via-Points Next	r (cm)	χ	Cyli Λ	inder Parameters ^B T
Lateral Deltoid	Humerus	2	Insertion	1.27	N.W.	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Anterior Deltoid	Humerus	2	Insertion	1.27	N.W.	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Posterior Deltoid	Humerus	3	Insertion	1.27	N.W.	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.0000
Supraspinatus	Humerus	3	Insertion	1.97	N.W.	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Infraspinatus	Humerus	2	Insertion	2.27	Positive	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Teres Minor	Humerus	2	Insertion	1.73	Positive	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Teres Major	Humerus	2	Insertion	2.27	Positive	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Pectoralis Major	Humerus	2	Insertion	1.5	N.W.	0	1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.1400 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 0.0000
Biceps Brachii	Forearm	3	Insertion	1.5	N.W.	0	0.0000 -1.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.0000
Triceps Brachii	Forearm	2	Insertion	1.5	N.W.	0	0.0000 -1.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 1.0000 0.0000 0.0000 0.0000 1.0000
Brachialis	Forearm	1	Insertion	1.5	N.W.	0	0.0000 -1.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 1.0000 1.0000 0.0000 0.0000 0.0000 1.0000

Table C.8: Parameters of muscle kinematics model II. Wrapping cylinders.

All transforms are given relative to the visual node transform of the COLLADA skeleton model (before the correction of the center of mass by the mass frame).

Table C.9: Parameters of muscle kinematics model III. The muscle kinematics model I
is extended by the following mesh wrapping surfaces. For the elbow joint muscles, the
muscle kinematics model III is identical to model II.

	Tendon		Ν	1esh Wrappin	g Surf	ace		
Muscle	l_{T_0}	В	Between	Via-Points	1	Meshe	s	${}^{S}\chi^{T}$
	(m)		Previous	Next	1	2	3	ĸ
Lateral Deltoid	0.3290	Humerus	2	Insertion	•	-	-	
Anterior Deltoid	0.2502	Humerus	2	Insertion	•	-	-	
Posterior Deltoid	0.2784	Humerus	2	Insertion	•	•	-	$\left \begin{array}{ccc} 0 & 0 & -1 \end{array} \right $
Supraspinatus	0.2197	Base	2	Insertion	-	-	•	$\left \begin{array}{ccc} 0 & 0 & -1 \end{array} \right $
Infraspinatus	0.2164	Humerus	2	Insertion	•	-	-	$\left \begin{array}{ccc} 0 & 0 & -1 \end{array} \right $
Teres Minor	0.2095	Humerus	2	Insertion	•	-	-	$\left \begin{array}{ccc} 0 & 0.5 & -1 \end{array} \right $
Teres Major	0.2170	Humerus	2	Insertion	•	•	-	
Pectoralis Major	0.1862	Humerus	2	Insertion	•	-	-	

The wrapping direction vectors are given relative to the visual node transform of the corresponding rigid body of the muscle insertion (see COLLADA skeleton model).

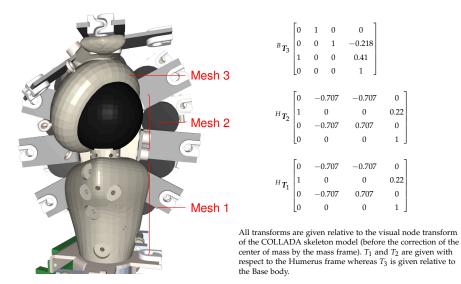


Figure C.2: Wrapping meshes of muscle kinematics model III.

C.3 SIMULATION-REALITY GAP

Table C.10: Glenohumeral joint equilibria distances. The table lists the mutual angular distances of the 17 glenohumeral joint equilibria in degrees.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.00	10.35	21.24	42.07	26.19	37.38	49.09	18.69	30.82	16.10	45.57	48.57	6.33	28.53	38.56	45.16	48.07
1		0.00	11.77	34.23	18.48	29.05	40.63	8.92	21.08	9.05	36.64	39.91	11.84	21.59	31.57	35.46	39.64
2			0.00	23.00	17.92	27.27	30.20	10.58	17.45	15.64	33.00	37.31	22.88	24.31	32.32	30.28	37.59
3				0.00	34.05	40.62	20.71	32.84	33.29	38.34	42.51	48.54	44.64	43.58	47.84	39.04	49.76
4					0.00	14.10	43.69	15.10	17.02	19.23	22.09	29.60	28.43	14.61	18.31	27.01	30.81
5						0.00	45.70	22.12	15.27	25.45	8.87	16.56	37.76	12.33	7.66	17.25	18.41
6							0.00	36.74	33.86	40.90	45.53	46.61	48.93	46.98	52.06	36.11	46.78
7								0.00	12.23	6.21	28.91	31.35	18.29	15.25	25.14	26.63	31.06
8									0.00	16.06	19.12	20.26	29.85	13.38	19.51	14.41	20.20
9										0.00	32.52	33.47	13.95	15.79	26.78	30.02	32.64
10											0.00	11.64	45.62	19.73	12.78	13.36	14.66
11												0.00	47.04	21.35	16.61	10.55	3.42
12													0.00	27.58	38.30	43.96	46.10
13														0.00	11.17	22.09	21.18
14															0.00	21.12	17.77
15																0.00	11.42
16																	0.00

 Table C.11: Complete robot equilibria distances. The table lists the mutual angular distances of the 6 robot equilibria in degrees.

			Glenohu	meral Joir	nt				Elbo	w Joint		
	E ₀	E_1	E_2	E_3	E_4	E_5	E ₀	E_1	E_2	E ₃	E_4	E_5
E_0	0.00	1.16	42.42	43.01	49.04	48.88	0.00	111.33	6.93	109.09	1.15	108.58
E_1		0.00	41.53	42.11	48.71	48.54		0.00	104.39	2.23	112.47	2.75
E_2			0.00	0.67	43.56	43.31			0.00	102.16	8.08	101.64
E_3				0.00	44.07	43.82				0.00	110.24	0.52
E_4					0.00	0.25					0.00	109.72
E_5						0.00						0.00

Table C.12: Pre-calibration simulation-reality gap summary. The table lists the average joint position and force errors of all three Anthrob models before calibration.

		Tie	er 1				Tie	r 2		
Model	Stati	cs	Dynar	nics		Statics		Ι	Dynamics	
Widdel	$\ \triangle \boldsymbol{q}_G \ $	$\overline{ \Delta f }$	$\ \triangle q_G \ $	$\overline{ \Delta f }$	$\ \triangle q_G \ $	$ \triangle q_E $	$\overline{ \Delta f }$	$\ \triangle \boldsymbol{q}_G \ $	$ \triangle q_E $	$\overline{ \Delta f }$
	(°)	(N)	(°)	(N)	(°)	(°)	(N)	(°)	(°)	(N)
I	21.9	5.0	15.2	4.4	14.3	4.4	3.1	8.7	5.1	2.9
II	15.6	5.4	11.5	4.3	13.0	5.1	2.7	7.5	5.8	2.7
III	11.3	5.3	8.7	4.1	10.4	4.8	2.7	6.3	5.6	2.7

D

ANTHROB CALIBRATION

D.1 SIMULATION-REALITY GAP

		Tie	er 1				Tie	r 2		
Model	Stati	cs	Dynar	nics		Statics		Γ	Dynamics	
Widder	$\ \triangle q_G \ $	$\overline{ \Delta f }$	$\ \triangle q_G \ $	$\overline{ \Delta f }$	$\ \triangle q_G \ $	$ \triangle q_E $	$ \Delta f $	$\ \triangle \boldsymbol{q}_G \ $	$ \triangle q_E $	$\overline{ \Delta f }$
	(°)	(N)	(°)	(N)	(°)	(°)	(N)	(°)	(°)	(N)
Ι	4.5	6.0	9.9	4.0	6.7	2.4	3.3	7.1	2.2	2.9
II	3.7	5.5	6.6	3.7	4.1	1.2	3.0	4.0	2.5	2.7
III	4.3	5.2	6.6	3.6	5.9	1.8	3.3	5.0	1.7	2.6

Table D.1: Post-calibration simulation-reality gap summary. The table lists the average joint position and force errors of all three Anthrob models after calibration.

D.2 CALIBRATION PARAMETERS

The tables on the following pages list the object and strategy parameter sets used for the calibration of the three Anthrob models.

$\begin{bmatrix} 1 \\ 1 \\ 0.4500 \end{bmatrix}$ $\begin{bmatrix} 1 \\ 0.0430 \\ 1 \\ 0.0430 \\ 1 \\ 0.4308 \end{bmatrix}$ $\begin{bmatrix} 1 \\ 0.04308 \\ 0.4270 \\ 0.4270 \\ 0.4270 \\ 0.4270 \\ 1 \\ 0.04270 \\ 1 \\ 0.04270 \\ 1 \\ 0.04270 \\ 0.0428 \\ 1 \\ 0.0528 \\ 1 \\ 0.0528 \\ 1 \\ 0.0528 \\ 1 \\ 0.0528 \\ 1 \\ 0.0528 \\ 1 \\ 0.0528 \\ 1 \\ 0.0528 \\ 0.0528 \\ 1 \\ 0.0528 \\ 0.0528 \\ 0.0528 \\ 0.0528 \\ 1 \\ 0.0528 \\ 0.052$
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
$ \begin{bmatrix} 1 \\ 1 \\ -0.0528 \\ 1 \\ -0.0528 \\ -0.0541 \\ -0.1720 \\ -0.1720 \\ -0.1723 \\ -0.1723 \\ -0.1723 \\ -0.1723 \\ -0.0402 \\ -0.0418 \\ -0.0402 \\ -0.0418 \\ -0.01710 \\ -0.0486 \\ -0.0418 \\ -0.0186 \\ -0.0418 \\ -0.0186 \\ -0.0528 \\ -0.1686 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.1549 \\ -0.25 \\ -0$
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$

		Table D	.3: Ob	ject par	ameters	of Anthro	D.3: Object parameters of Anthrob model I (tier 1, dynamics).	(tier 1, (dynami	cs).							
Tier	Stage	Muscle			Muscle Kinematics	nematics						Actuator	tor				
					Via-Point p	int <i>p</i>			Spin	Spindle radius r_S	rs			Tend	Tendon length $l_{T_{\rm 0}}$	l_{T_0}	
			Idx	Range	σ	Initial	Final	Min.	Max.	σ	Initial	Final	Min.	Max.	σ	Initial	Final
			(1)	(mm)	(mm)	(m)	(m)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
				[±2.5]	0.5	0.0015	0.0006										
1	Dynamics	Lateral Deltoid	2	土2.5	0.5	-0.258	-0.2606	6.5	8.5	0.25	7.67	7.64	0	1000	1	348.5	350.2
				±2.5	0.5]	0.450	0.4499										
				[±2.5]	0.5	0.045	0.0477										
		Anterior Deltoid	2	土2.5	0.5	-0.153	-0.1532	6.5	8.5	0.25	6.51	6.52	0	1000	1	249.9	251.8
				±2.5	0.5	0.431	0.4326										
				[±2.5]	0.5	-0.0355	-0.0339										
		Posterior Deltoid	2	±2.5	0.5	-0.2229	-0.2223	6.5	8.5	0.25	7.08	6.64	0	1000	1	281.6	282.5
				±2.5	0.5	0.4249	0.4256										
				[±2.5]	0.5	[-0.0492]	[-0.0479]										
		Supraspinatus	2	±2.5	0.5	-0.1698	-0.1674	6.3	7.5	0.125	7.09	6.35	0	1000	1	225.8	226.5
				±2.5	0.5]	0.4190	0.4191										
				[±2.5]	0.5	[-0.0541]	-0.0553										
		Infraspinatus	2	±2.5	0.5	-0.1723	-0.1727	6.3	7.5	0.125	6.93	6.41	0	1000	1	218.8	216.9
				±2.5]	0.5]	0.3838	0.3813										
				[±2.5]	0.5	[-0.0418]	[-0.0427]										
		Teres Minor	7	土2.5	0.5	-0.1686	-0.1677	6.3	7.5	0.125	6.76	6.34	0	1000	1	202.7	206.7
				±2.5]	0.5]	0.3465	0.3466										
				[±2.5]	0.5	0.0517	0.0527										
		Teres Major	7	土2.5	0.5	-0.1549	-0.1528	6.3	7.5	0.125	6.43	6.32	0	1000	1	220.4	222.4
				±2.5	0.5]	0.3840	0.3859										
				±2.5	0.5	0.0390	0.0392										
		Pectoralis Major	7	± 2.5	0.5	-0.1539	-0.1552	6.3	7.5	0.125	6.73	7.14	0	1000	1	184.8	184.9
				[±2.5]	0.5]	0.3501	0.3511										

Table D.3: Object parameters of Anthrob model I (tier 1, dynamics).

						Table	D.4: (Object p	aramete	ers of Antr	Table D.4: Object parameters of Anthrob model I (tier 2)	II (tier	2)								
Stage	Muscle					Muscle Kinematics	Sinemati	SS								Actı	Actuator				
				Via-P	Via-Point p				Via-P	Via-Point <i>p</i>			Spi	Spindle radius r_S	us r _S			Tenc	Tendon length l_{T_0}	$1 I_{T_0}$	
		Idx	Range	q	Initial	Final	Idx	Range	q	Initial	Final	Min.	Max.	σ	Initial	Final	Min.	Max.	σ Initial	Initial	Final
		(1)	(mm)	(mm)	(m)	(m)	(1)	(mm)	(mm)	(m)	(m)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
			±2.5	[1]	0.0271	0.0294		±2.5	[1]	-0.017	-0.0170										
Statics	Biceps	2	± 2.5	1	-0.2052	-0.2074	з	± 2.5	1	0.139	0.1391	6.5	8.5	0.5	6.5	8.16	0	1000	2	470.8	485.1
	Brachii		±2.5	1	0.4503	0.4522		±2.5	1	0.017	0.0204										
			±2.5	1	0.0000	-0.0005															
	Triceps	2	± 2.5	1	0.0470	0.0459	I	I	I	I	I	6.5	8.5	0.5	6.5	6.83	0	1000	2	128.4	129.7
	brachil		_±2.5	1	[-0.0300]	[-0.0304]															
			±2.5	1	0.0089	0.0082															
	Brachialis	1	± 2.5	1	0.1090	0.1105	I	I	I	I	I	6.5	8.5	0.5	6.5	6.55	0	1000	2	170.3	169.8
			[±2.5]	1	0.0358	0.0370															
			±2.5	0.5	0.0294	0.0304		±2.5	0.5	-0.0170	-0.0167										
Dynamics	Biceps	2	± 2.5	0.5	-0.2074	-0.2099	ω	± 2.5	0.5	0.1391	0.1403	6.5	8.5	0.25	8.16	8.30	0	1000	1	485.1	486.7
	Brachu		±2.5	0.5	0.4522	0.4525		_±2.5_	0.5	0.0204	0.0233										
			±2.5	0.5	-0.0005	0.0011															
	Triceps	2	± 2.5	0.5	0.0459	0.0458	I	I	I	I	I	6.5	8.5	0.25	6.83	6.80	0	1000	1	129.7	128.7
	Brachu		±2.5	0.5	[-0.0304]	[-0.0321]															
			±2.5	0.5	0.0082	0.0085															
	Brachialis	1	± 2.5	0.5	0.1105	0.1107	I	I	I	I	I	6.5	8.5	0.25	6.55	7.00	0	1000	1	169.8	169.2
			±2.5	0.5	0.0370	0.0362															

						1 100[20						-	סומווייי									
Tier	Stage	Muscle				Mi	Muscle Kinematics	tics									Actuator	tor				
					Via-Point p	oint <i>p</i>			Cyliı	Cylinder radius r_C	lius r _C			Spin	Spindle radius r_S	s 15			Tendo	Tendon length l_{T_0}	l_{T_0}	
			Idx	Range	σ	Initial	Final	Min.	Мах.	σ	Initial	Final	Min.	Мах.	σ	Initial	Final	Min.	Мах.	σ	Initial	Final
			(1)	(mm)	(mm)	(m)	(m)	(cm)	(cm)	(cm)	(cm)	(cm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
				[±2.5]	1	000000	0.0023															
1	Statics	Lateral Deltoid	2	土2.5	1	-0.2570	-0.2554	0.5	1.27	0.1	1.27	0.527	6.5	8.5	0.5	6.5	7.05	0	1000	2	329.5	340.0
				土2.5	-	0.4500	0.4482															
				土2.5	1	0.0430	0.0410															
		Anterior Deltoid	7	土2.5	1	-0.1540	-0.1553	0.5	1.27	0.1	1.27	1.26	6.5	8.5	0.5	6.5	7.29	0	1000	5	245.2	253.7
				±2.5	1	0.4308	0.4331															
				±2.5	1	-0.0250	0.0274															
		Posterior Deltoid	С	± 2.5	1	-0.2350	-0.2374	0.5	1.27	0.1	1.27	1.03	6.5	8.5	0.5	6.5	6.53	0	1000	2	280.9	280.5
				±2.5	-	0.4100	0.4075															
				[±2.5]	1	[-0.0280]	[-0.0291]															
		Supraspinatus	ю	土2.5	1	-0.2250	-0.2248	0.5	1.97	0.1	1.97	1.96	6.3	7.5	0.25	6.3	6.90	0	1000	2	223.9	232.2
				±2.5	[]	0.4000	0.4014															
				±2.5	1	-0.0528	-0.0549															
		Infraspinatus	2	± 2.5	1	-0.1720	-0.1742	0.5	2.27	0.1	2.27	2.22	6.3	7.5	0.25	6.3	6.33	0	1000	2	217.0	219.3
				±2.5	-	0.3837	0.3823															
				±2.5	1	-0.0403	-0.0396															
		Teres Minor	7	± 2.5	1	-0.1710	-0.1724	0.5	1.73	0.1	1.73	1.27	6.3	7.5	0.25	6.3	6.45	0	1000	2	208.9	203.4
				±2.5	-	0.3485	0.3484															
				± 2.5	1	0.0528	0.0541															
		Teres Major	7	土2.5	1	-0.1540	-0.1521	0.5	2.27	0.1	2.27	1.80	6.3	7.5	0.25	6.3	6.645	0	1000	2	217.9	221.2
				±2.5	1	0.3837	0.3829															
				±2.5	1	0.0402	0.0415															
		Pectoralis Major	7	土2.5	1	-0.1540	-0.1543	0.5	1.5	0.1	1.5	1.43	6.3	7.5	0.25	6.3	6.75	0	1000	5	187.0	186.2
				±2.5]	[1]	0.3485	0.3463	_									_					

Table D.5: Object parameters of Anthrob model II (tier 1, statics).

				100					000													
Tier	Stage	Muscle				Mı	Muscle Kinematics	ics									Act	Actuator				
					Via-Point <i>p</i>	int p			Cyli	Cylinder radius r_C	ius r _C			Spii	Spindle radius r_S	us r _S			Tenc	Tendon length l_{T_0}	$h l_{T_0}$	
			Idx	Range	q	Initial	Final	Min.	Max.	σ	Initial	Final	Min.	Max.	σ	Initial	Final	Min.	Max.	q	Initial	Final
			(I)	(mm)	(mm)	(m)	(m)	(cm)	(cm)	(cm)	(cm)	(cm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
				±2.5	0.5	0.0023	0.0029															
1	Dynamics	Lateral Deltoid	2	± 2.5	0.5	-0.2554	-0.2570	0.5	1.27	0.05	0.527	0.674	6.5	8.5	0.25	7.05	6.50	0	1000	1	340.0	338.2
				±2.5	0.5	0.4482	0.4483															
				±2.5	0.5	0.0410	0.0395															
		Anterior Deltoid	2	± 2.5	0.5	-0.1553	-0.1529	0.5	1.27	0.05	1.26	1.27	6.5	8.5	0.25	7.29	6.55	0	1000	1	253.7	249.9
				_±2.5	0.5	0.4331	0.4310															
				±2.5	0.5	0.0274	-0.0275															
		Posterior Deltoid	ω	± 2.5	0.5	-0.2374	-0.2369	0.5	1.27	0.05	1.03	1.19	6.5	8.5	0.5	6.53	6.51	0	1000	1	280.5	280.9
				_±2.5	0.25	0.4075	0.4053															
				±2.5	0.5	-0.0291	-0.0296															
		Supraspinatus	ω	± 2.5	0.5	-0.2248	-0.2244	0.5	1.97	0.05	1.96	1.56	6.3	7.5	0.125	6.90	7.35	0	1000	1	232.2	234.5
				±2.5	0.5	0.4014	0.4032															
				±2.5	0.5	-0.0549	-0.0532															
		Infraspinatus	2	± 2.5	0.5	-0.1742	-0.1741	0.5	2.27	0.05	2.22	2.17	6.3	7.5	0.125	6.33	6.30	0	1000	1	219.3	217.2
				_±2.5	0.5	0.3823	0.3801															
				±2.5	0.5	-0.0396	-0.0392															
		Teres Minor	2	± 2.5	0.5	-0.1724	-0.1721	0.5	1.73	0.05	1.27	1.45	6.3	7.5	0.125	6.45	6.33	0	1000	1	203.4	206.6
				_±2.5	0.5	0.3484	0.3481															
				±2.5	0.5	0.0541	0.0547															
		Teres Major	2	± 2.5	0.5	-0.1521	-0.1522	0.5	2.27	0.05	2.27	2.13	6.3	7.5	0.125	6.645	6.58	0	1000	1	221.2	220.1
				±2.5	0.5	0.3829	0.3815															
				±2.5	0.5	0.0415	0.0424															
		Pectoralis Major	2	± 2.5	0.5	-0.1543	-0.1559	0.5	1.5	0.05	1.43	1.48	6.3	7.5	0.125	6.75	6.82	0	1000	1	186.2	185.6
				± 2.5	0.5	0.3463	0.3468															

 * if any final value exceeds the defined value bounds than this is due to round-of errors

ANTHROB CALIBRATION

Stage	Muscle							Muscle Kinematics	Kinem	atics											Actuator	tor			
				Via-Point p	nt <i>p</i>				Via-	Via-Point <i>p</i>			Cylin	Cylinder radius <i>r</i> _C	ius r_C			Spindle	Spindle radius r_S	s r.s			Tendor	Tendon length l_{T_0}	l_{T_0}
		(I)	Idx Range (1) (mm)	σ I (mm)	Initial (m)	Final (m)	Idx Range (1) (mm)		σ (mm)	Initial (m)	Final (m)	Min. (cm)	Min. Max. σ (cm) (cm) (cm)	σ (cm)	Min. Max. σ Initial Final Min. Max. (cm) (cm) (cm) (cm) (cm) (mn) (mm) (mm) (mm) (mm) (mm) (mm) (m	Final [Min.] (i)	Max. 1) (i	σ I (i)	nitial (σ Initial Final Min. Max. mm (mm) (mm) (mm) (mm)	Min. (mm)	Max. mm) (i	Final Min. Max. σ Initial Final Min. Max. σ Initial Final (min. (mm) (mm) (mm) (mm) (mm) (mm) (mm) (mm	Initial Final (mm) (mm)
					0.0271	0.0278			Ē	-0.0170	-0.0171	<u>)</u>				<u> </u>					<u> </u>				
Statics	Biceps Brachii	7	土2.5		-0.2052	-0.2062	с	土2.5	1	0.1390	0.1406	1	2	0.1	1.5	1.99	6.5	8.5	0.5	6.5	7.35	0	1000	2	471.5 470.3
			[±2.5] [±2.5]	 	0.4503] [0.0000]	[0.4479] [-0.0022]	_	±2.5]	1	0.0170	0.0191														
	Triceps Brachii	2	土2.5	1 0	0.0470	0.0457	I	I	I	I	I	1	2	0.1	1.5	1.96	6.5	8.5	0.5	6.5	6.52	0	1000	2	128.4 128.7
			±2.5		-0.0300	0.0312																			
			±2.5	 	0.0089	0.0108																			
	Brachialis	-	±2.5	1 0	0.1090	0.1115	I	I	I	I	I	-	7	0.1	1.5	1.93	6.5	8.5	0.5	6.5	7.73	0	1000	2	170.3 177.3
			±2.5	1 [0	0.0358	0.0344										—					—				
			±2.5	 [1] [0	0.0278	0.0255	_	±2.5	1	$\begin{bmatrix} -0.0171 \end{bmatrix}$	[-0.0168]														
Dynamics	Biceps Brachii	7	土2.5	T	-0.2062	-0.2081	б	±2.5	1	0.1406	0.1405	1	7	0.1	1.99	1.86	6.5	8.5	0.5	7.35	7.49	0	1000	2	470.3 475.1
			±2.5		0.4479	0.4486		±2.5	1	0.0191	0.0204														
			± 2.5	 ī 	-0.0022	-0.0032																			
	Triceps Brachii	7	±2.5	1 0	0.0457	0.0451	I	I	I	I	I		7	0.1	1.96	1.70	6.5	8.5	0.5	6.52	7.33	0	1000	2	128.7 127.7
			[±2.5]	 ' :	0.0312																				
			± 2.5	1	0.0108	0.0135																			
	Brachialis	1	± 2.5	1 0	0.1115	0.1097	I	I	I	I	I	1	7	0.1	1.93	1.24	6.5	8.5	0.5	7.73	7.1	0	1000	2	177.3 173.0
			± 2.5	1	0.0344	0.0347																			

Teres Major 2		Teres Minor 2	Infraspinatus 2	Supraspinatus 3	Posterior Deltoid 3	Anterior Deltoid 2	1 Statics Lateral Deltoid 2	Idx (1)	Tier Stage Muscle	Table D.8: Object parameters of Anthrob model III (tier 1, statics).
±2.5 ±2.5	(+2.5 (+2.5) (+2.5)	+ + 2.5 5 5	[#2.5] [#2.5] [#2.5] [#2.5]				±2.5 ±2.5 ±3.5	Range (mm) (r	Mi	ect parar
<u> </u>					·			Via-Point p σ In(mm)(Muscle Kinematics	neters (
0.0402 -0.1540	0.0528 -0.1540 0.3837	-0.0402 -0.1710	0.4189 -0.0528 -0.1720 0.3837	-0.0485	-0.0351 -0.2246 0.4270	0.0430 -0.1540	0.0000	t p Initial (m)	matics	of Anthr
0.0403	$\begin{bmatrix} 0.0540 \\ -0.1521 \\ 0.3828 \end{bmatrix}$	$\begin{bmatrix} -0.0406 \\ -0.1689 \\ 0.3487 \end{bmatrix}$	$\begin{bmatrix} -0.0021\\ -0.0528\\ -0.1731\\ 0.3831 \end{bmatrix}$	-0.0009	0.0018 0.0948 -0.0013	0.0429	-0.0024 -0.2568 0.4492	Final (m)		ob model I
6.3	6.3	6.3	6.3	6.3	6.5	6.5	6.5	Min. (mm)		II (tier 1
7.5	7.5	7.5	7.5	7.5	8.5	8.5	8.5	Spin Max. (mm)		l, static
0.25	0.25	0.25	0.25	0.25	0.5	0.5	0.5	Spindle radius <i>r_S</i> . σ In .) (mm) (n		s).
6.3	6.3	6.3	6.3	6.3	6.5	6.5	6.5	nitial (mm)		
6.66	7.15	7.13	6.37	6.73	6.66	6.58	6.59	Final (mm)	Actuator	
0	0	0	0	0	0	0	0	Min. (mm)	tor	
1000	1000	1000	1000	1000	1000	1000	1000	Tendon Max. (mm) (i		
2 18	2	2 20	2	2 2	2	2 2	2 3	Tendon length <i>l</i> T ₀ ιx. σ In n) (mm) (n		
186.2 185.8	217.0 221.3	209.5 230.9	216.4 214.0	219.7 225.9	278.4 279.4	250.2 250.6	329.1 339.7	^{1 I} T ₀ Initial Final (mm) (mm)		

 * if any final value exceeds the defined value bounds than this is due to round-of errors

174 ANTHROB CALIBRATION

		lable D.S		ect para	meters	.9: Object parameters of Anthrob model III (tier 1, dynamics)	model III	(tier 1,	dynam	cs).							
Tier	Stage	Muscle			Muscle Kinematics	nematics						Actuator	or				
					Via-Point p	int <i>p</i>			Spin	Spindle radius r_S	S			Tendo	Tendon length l_{T_0}	T_0	
			Idx	Range	σ	Initial	Final	Min.	Max.	σ	Initial	Final	Min.	Мах.	σ	Initial	Final
			(1)	(mm)	(uuu)	(m)	(m)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
				[±2.5]	1	[-0.0024]	0.0001										
1	Dynamics	Lateral Deltoid	2	土2.5	1	-0.2568	-0.2553	6.5	8.5	0.5	6.59	6.65	0	1000	7	339.7	338.0
				±2.5	1	0.4492	0.4511										
				±2.5	1	0.0429	0.0418										
		Anterior Deltoid	2	土2.5	1	-0.1549	-0.1527	6.5	8.5	0.5	6.58	7.07	0	1000	2	250.6	248.8
				±2.5	1	0.4294	0.4291										
				±2.5	1	0.0018	0.0019										
		Posterior Deltoid	ю	土2.5	1	0.0948	0.0949	6.5	8.5	0.5	6.66	6.67	0	1000	7	279.4	277.9
				±2.5	1	-0.0013	0.0000										
				±2.5	1	-00000	-0.0003										
		Supraspinatus	ю	±2.5	1	-0.0006	-0.0014	6.3	7.5	0.25	6.73	6.73	0	1000	7	225.9	228.2
				±2.5	1	-0.0021	-0.0001										
				±2.5	1	-0.0528	[-0.0521]										
		Infraspinatus	2	±2.5	1	-0.1731	-0.1716	6.3	7.5	0.25	6.37	6.58	0	1000	2	214.0	213.1
				±2.5	-	0.3831	0.3833										
				± 2.5		-0.0406	-0.0430										
		Teres Minor	2	土2.5	1	-0.1689	-0.1699	6.3	7.5	0.25	7.13	6.78	0	1000	7	230.9	222.1
				±2.5	1	0.3487	0.3499										
				±2.5	-	0.0540	0.0560										
		Teres Major	2	±2.5	1	-0.1521	-0.1535	6.3	7.5	0.25	7.15	7.17	0	1000	2	221.3	221.1
				±2.5	1	0.3828	0.3825										
				±2.5		0.0403	0.0386										
		Pectoralis Major	2	土2.5	1	-0.1531	-0.1512	6.3	7.5	0.25	6.66	7.19	0	1000	7	185.8	180.5
				±2.5	[1]	0.3502	0.3487										

Table D.9: Object parameters of Anthrob model III (tier 1, dynamics).

					Ъ	ble D.10	000	oject p	oaran	neters of	Table D.10: Object parameters of Anthrob model III (tier	moc	del III	l (tier	2)											
Stage	Muscle							Muscle	Muscle Kinematics	natics											Act	Actuator				
				Vi	Via-Point p				Via	Via-Point p			Cylin	Cylinder radius r _C	lius r _C			Spin	Spindle radius r_S	lius r _S			Ten	don ler	Tendon length l_{T_0}	
		Idx	Idx Range	e q	Initial	Final	Idx	Idx Range	σ	Initial	Final	Min.	Min. Max. σ	q	Initial	Final	Min.	Max.	σ	Initia	l Final	Min	. Max	α	Initia	nitial Final Min. Max. σ Initial Final Min. Max. σ Initial Final
		(1)		(mm) (mm)) (m)	(m)	(1)	(mm)	(mm) (mm)	(m)	(m)	(cm)	(cm) (cm) (cm)	(cm)	(cm)	(cm)	(mm)	(mm)	(mm)	(mm)	(mm)) (mm	i) (mm) (mm	i) (mm	(cm) (cm) (mm) (mm) (mm) (mm) (mm) (mm)
			±2.5	[1]	0.0271	0.0274	_	±2.5	1	-0.0170	-0.0149	_														
Statics	Biceps Brachii	2	±2.5	1	-0.2052	-0.2038	ω	± 2.5	1	0.1390	0.1368	11	2	0.1	1.5	1.51	6.5	8.5	0.5	6.5	7.75	0	1000		471.	2 471.5 474.2
			±2.5	Ξ	0.4503	0.4493		±2.5	1	0.0170	0.0176															
			±2.5	-	0.0000	-0.0021																				
	Triceps Brachii	2	±2.5	1	0.0470	0.0451	I	I	I	I	I	1	2	0.1	1.5	1.48	6.5	8.5	0.5	6.5	6.94	0	1000	2	128.4	4 128.0
			±2.5	Ξ	[-0.0300]	[-0.0293]																				
			±2.5	1	0.0089	0.0088																				
	Brachialis	-	±2.5	1	0.1090	0.1068	I	I	I	I	I	1	2	0.1	1.5	1.67	6.5	8.5	0.5	6.5	6.59	0	1000	2		170.3 166.9
			[±2.5]	Ξ	0.0358	0.0338																				
			±2.5	1	0.0274	0.0270		±2.5	1	-0.0149	-0.0143															
Dynamics	Biceps Brachii	2	±2.5	1	-0.2038	-0.2048	ω	± 2.5	1	0.1368	0.1360	1	2	0.1	1.51	1.44	6.5	8.5	0.5	7.75	7.96	0	1000	2		474.2 475.3
			±2.5	Ξ	0.4493	0.4478		±2.5	1	0.0176	0.0198															
			±2.5	1	-0.0021	-0.0043																				
	Triceps Brachii	2	±2.5	1	0.0451	0.0428	I	I	I	I	I	1	2	0.1	1.48	1.78	6.5	8.5	0.5	6.94	7.61	0	1000	2		128.0 128.7
			±2.5	1	[-0.0293]	[-0.0305]																				
			±2.5	1	0.0088	0.0083																				
	Brachialis	1	±2.5	1	0.1068	0.1087	I	I	I	I	I	1	2	0.1	1.67	1.69	6.5	8.5	0.5	6.59	6.62	0	1000	2		166.9 170.8
			±2.5]	0.0338	0.0363																				

Table D.11: Joint friction object parameters. For the dynamics stages, the joint friction parameters were included in the set of object parameters.

Joint Friction	Final Min. Max. σ Initial Final Nm) (% of τ_5) (Nms/rad) (Nms/rad) (Nms/rad)		00024 0.0 100 10 0.0 20.8 0.0 0.4 0.025 0.15 0.333	0005 0.0 100 10 0.0 2.5 0.0 0.4 0.025 0.15 0.057	.0013 0.0 100 10 0.0 41.4 0.0 0.4 0.025 0.15 0.109		.0724 0.0 100 10 0.0 44.8 0.0 0.4 0.04 0.02 0.0440	1.1847 0.0 100 10 0.0 1.0 0.0 0.4 0.02 0.0285	.098 0.0 100 10 0.0 8.4 0.0 0.4 0.04 0.02 0.0429
	Final (N m)		0.00024	0.0005	0.0013		0.0724	0.1847	0.0988
	^{ts} Max. σ Initial (Nm) (Nm) (Nm)	1	0.2 0.025 0.0	0.2 0.025 0.0	0.2 0.025 0.0	1	0.2 0.05 0.0	0.2 0.05 0.0	0.2 0.05 0.0
		ı 	0.0	0.0	0.0	I	0.0	0.0	0.0
Joint	Glenohumeral Elbow Min.	1	•	•	•	1	•	•	•
Muscle Kine- matics Model		П/П/I	I	Π	Ш	П1/П/I	I	п	Ш
Stage		Statics		Dynamics		Statics		Dynamics	
Tier		1		1		7		2	

E

CALIPER: A ROBOTICS DEVELOPMENT ENVIRONMENT

A robotics software framework, called CALIPER, has been developed which is particularly tailored for the simulation of tendon-driven robots. The framework is implemented in C++ and relies on Qt as well as on CMake as build system for platform independence. In itself, CALIPER provides only some very basic built-in functionality, such as a logging system or a container for Graphical User Interfaces (GUIs). However, it can be extended at runtime by *plugins* which makes it a highly versatile and easy-to-use framework (see Figure E.1). CALIPER is already in use by leading research institutions in Europe.

E.1 CALIPER CORE

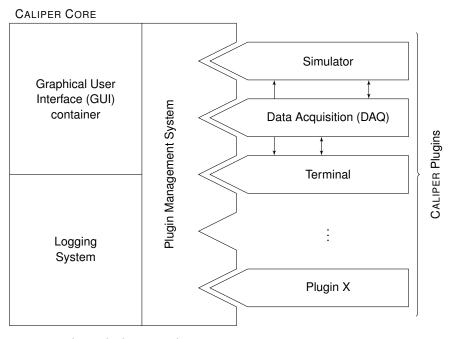
The CALIPER core consists of three components (see Figure E.1): (i) a *Overview* GUI, derived from QMainWindow, which provides a basic user-interface for changing application settings and which serves as a container for the GUIs of the plugins (see below), (ii) a logging system based on the C++ logging library log4cpp and, most importantly, a plugin management system which provides a GUI for activating and deactivating plugins at runtime as well as for configuring plugin search folders.

The CALIPER GUI exploits Qt's dock widget concept (see QDockWidget) to provide a flexible user-interface that can be adjusted to the particular needs of an experiment and/or user. In this concept, the application window is divided into five "dock areas" and a plugin can be either docked to one of these areas or be floating as top-level window on the desktop. The current arrangement of the GUI (size, position and used docking area of each plugin) is stored at application exit and restored at restart using the platform independent helper class QSettings. Furthermore, a settings dialog is provided which can be extended by plugins at runtime to include plugin-specific settings.

Log files are inevitable to debug application problems, particular in *Logging system* an extensible framework in which plugins from different developers are interacting. Therefore, a logging facility was integrated into CALIPER based on the log4cpp library. The logger can be configured

Graphical User

Interface (GUI)



Inter-plugin connection

Figure E.1: CALIPER framework overview. The software framework CALIPER has been developed to simulate the dynamics of tendon-driven robots. In itself, the framework provides only basic functionality (such as a logging system) but new functions can be added at runtime via *plugins*.

via the settings dialog (see Figure E.3) and supports five different priority levels for the log messages (Debug, Info, Warn, Error and Fatal).

Plugin management The most important part of the CALIPER core, is the plugin management ment system. It consists of a GUI for configuring plugin search folders, i. e. folders that will be scanned recursively for valid CALIPER plugins (see Figure E.2a) and an interface for activating/deactivating detected plugins at runtime (see Figure E.2b). Moreover, it ensures that plugin interconnections are established (if required) and that plugin settings are properly stored to and restored from a central application settings file.

E.2 CALIPER PLUGINS

CALIPER plugins are implemented via the Qt Plugin API, which provides a platform-independent mechanism for extending applications at runtime. By default, CALIPER ships with three plugins: (i) a simulator plugin which facilitates the physics-based simulation of robots and/or other dynamic systems, (ii) a data acquisition plugin which provides tools for analyzing simulation model parameters in real-time and (iii) a terminal plugin which enables user interactions with the simulator (see also Figure E.3).

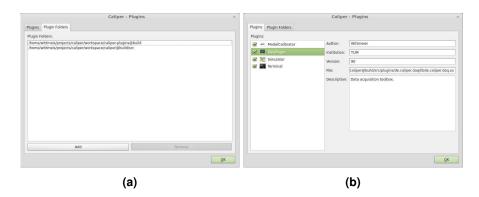


Figure E.2: CALIPER plugin management system. In CALIPER, plugins are used to provide and extend the application functionality. **(a)** Multiple "plugin folders" can be configured that are recursively scanned for valid plugins at application startup. **(b)** Detected plugins can be activated and deactivated at runtime by the user.

The developed simulator plugin relies on the open-source physics engine Bullet Physics for simulating the dynamics of the kinematic chain of the robot (rigid body and constraint dynamics, respectively) and on the open-source graphics engine Coin3D for visualizing the simulation results. On top of the engines, an abstraction layer, called Scene-Graph Abstraction Layer (SGAL), has been implemented, which provides an engine-independent access to the simulation model for higher software layers. The abstraction layer is based on the Robotics Library developed by M. Rickert and currently supports the importing of model files specified in accordance with the open-standard XML schema Collaborative Design Activity (COLLADA) (see also [166]). However, by default, simulating the dynamics of muscle equivalents is not supported by Bullet Physics. Thus, an extension mechanism was implemented which enables the integration of different muscle models at runtime (similar to the plugins of the CALIPER core). By default, CALIPER ships with two muscle models: (i) a Hill muscle model [43] and (ii) an electromagnetic tendon-driven muscle model as described in the earlier chapters of this thesis. However, it is important to emphasize that this extension mechanism is not limited to the integration of muscle models but can actually be used for the implementation of any kind of model extension as long as the underlying dynamics can be modeled by applying forces to the rigid bodies of the Bullet model. Simulation models that require extensions are stored and loaded via a custom Extensible Markup Language (XML) file format, called Caliper Scene File (CSF).

To facilitate the analysis of the simulation results, a Data Acquisition (DAQ) plugin has been developed (see Figure E.3). The plugin supports the real-time visualization of an arbitrary number of parameters over time and provides tools for plot configuration, plot export (to SVG), data export and printing. Furthermore, the rate of the sampling can be user-adjusted and the recorded data can be plotted either Simulator plugin

Data Acquisition (DAQ) plugin

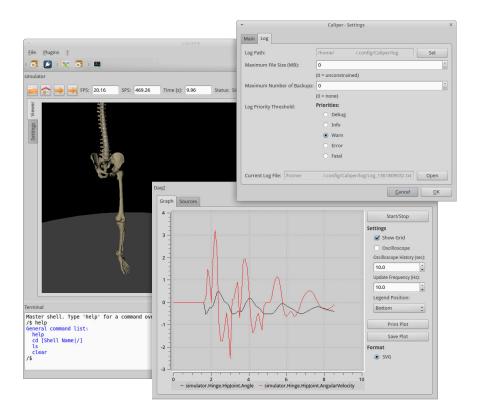


Figure E.3: CALIPER screenshots. By default, CALIPER ships with three plugins: (i) a simulator plugin (top left, showing a leg model), (ii) a terminal plugin for manually interfacing other plugins (bottom left) and (iii) a DAQ plugin for exporting, printing and analyzing model parameters in real-time (bottom right). Furthermore, a settings dialog is provided which can be extended by the plugins to consolidate all application settings (top right).

in *normal* or *oscilloscope* mode. While in normal mode the entire time history of the current recording is shown, the oscilloscope mode only displays the past *n* seconds. Theoretically, data sources of any type can be registered to the DAQ plugin but currently only scalars, vectors and quaternions are supported. The registered sources are listed in a tree-like fashion and can be selected individually by the user for recording and/or plotting.

Terminal plugin The third and last plugin that is part of the default CALIPER distribution is the terminal plugin, which is based on the QConsole widget developed by H. Bdioui (see Figure E.3). It provides an interface for registering and unregistering *shells* which can be accessed using the command cd—similar to the change directory command of the Linux terminal. Each registered shell provides a set of commands that can be evoked by the user to directly interact with individual CALIPER plugins.

QT PIPELINE TOOLKIT (QPT)

In computer science, two techniques are common to increase the performance of a task: (i) pipelining and (ii) parallelization [31]. While in pipelining a task is partitioned in a sequence of stages that are executed sequentially (either synchronously using a clock signal, or asynchronously using a hand-shake mechanism [57]), in parallelization multiple tasks or subtasks are executed simultaneously (e.g. on different CPUs). Here we present an open-source software framework that was developed for the efficient implementation and execution of asynchronous software pipelines. The framework, which is called Qt Pipeline Toolkit (QPT), is implemented in C++ and based on Qt. It uses a shared memory approach for inter-stage communication and CMake as build system. It is available for download at Sourceforge.

In QPT, a software pipeline is represented by an object of the class QptPipeline, which is derived from QThread and provides methods for adding and removing pipeline stages as well as for executing the pipeline stages in either the (i) non-threaded or (ii) threaded mode. In the non-threaded mode, a thread is started only for the pipeline object and each pipeline stage is executed in this thread in the order in which it was added to the pipeline. This is identical to the sequential execution of the pipeline stages and hence no performance increase due to pipelining occurs. In contrast, in threaded mode, a thread is started for each stage of the pipeline and not for the pipeline object. In this case, the pipeline execution is identical to an asynchronous pipeline (pipeline stages are not executed with respect to a central clock but the stages synchronize with each other via the data ports see below).

In QPT, a pipeline stage is created by instantiating objects of the class QptStage. Each QptStage is derived from QThread and can have an arbitrary number of data in- and outports that are registered to the stage for inter-stage communication (implemented by the classes QptInport and QptOutport, see below). The execution of each stage proceeds as follows. First, it checks whether all registered and connected QptInports have data. If this is not the case, the execution of the stage blocks, otherwise it proceeds with the execution by calling

Overview

The class QptPipeline

The class QptStage

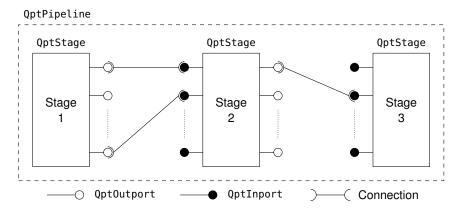


Figure F.1: Qt Pipeline Toolkit (QPT) overview. In QPT, a software pipeline is represented by an object of the class QptPipeline which consists of pipeline stages implemented by the class QptStage. For inter-stage communication, each QptStage can provide an arbitrary number of QptInports and QptOutports that are connected at compile-time. The QptPipeline can be executed either in the *non-threaded* (sequential) or *threaded* (parallel) mode, where each QptStage is executed in a dedicated thread.

the pure virtual method applyInternal(). Once applyInternal() returns, the processed data packages are forwarded to the next stage by calling the method pushData() on all registered QptOutports of the stage. Finally, the input data of each QptInport of the stage is marked as "processed" by calling the method dataProcessed() on all QptInports, which ensures the correct deallocation of the memory associated with the data (see below). In order to implement a pipeline stage and perform a specific task, the user has to derive from QptStage and implement the pure virtual method applyInternal().

As mentioned already, a stage provides two types of data ports: (i) Data ports and inter-stage inports that receive data from a previous stage in the pipeline (implecommunication mented by the class QptInport) and (ii) outports that forward data to the next stage in the pipeline (implemented by the class QptOutport). An outport can be connected to an arbitrary number of inports via the method connect(), which renders it possible to create parallel pipeline structures as shown in Figure A.2 of Appendix A. For the efficient exchange of data packages between in- and outports, a sharedmemory approach based on pointers was used. To prevent memory leaks, a reference counting system was implemented that ensures that the memory associated with a data package is only deallocated if all stages that require the data package have completed its processing. Furthermore, inports as well as outports are implemented as class templates which makes it possible to validate the data type conformity of connected data ports at compile-time.

Pipeline and stageAnalyzing the performance of pipelines and of parallel applicationsprofilingin general can be an extremely tedious task. Therefore, QPT includesmechanisms and classes that facilitate the profiling of the entire pipe-

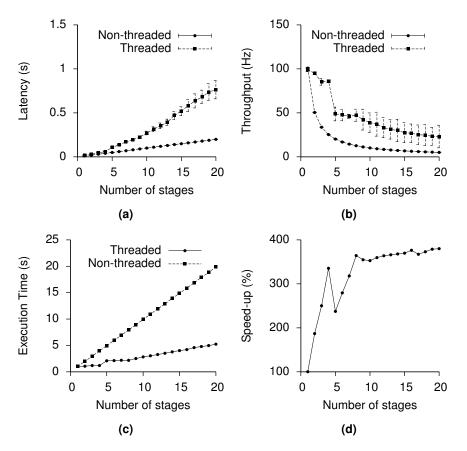


Figure F.2: Qt Pipeline Toolkit (QPT) benchmarking results. The performance of QPT was analyzed by computing 100 times the prime factors of all integers between 4 and 25000 by an increasing number of successively interconnected stages. **(a)** Pipeline latency. **(b)** Pipeline throughput. **(c)** Pipeline execution time. **(d)** Pipeline speed-up.

line and of individual stages. These are: (i) Qt signals defined in the classes QptPipeline and QptStage that can be used for performance benchmarks and (ii) pre-defined helper and profiler classes that facilitate the analysis of the timing parameters of pipelines and stages. For details please refer to the QPT documentation.

The performance of the developed framework was assessed by performing prime factorizations of integers. Thus, a pipeline stage, called PrimeFactorization, was implemented that computes the prime factors of integers using trial divisions [17]. For the benchmark, an increasing number of $1 \le n \le 20$ PrimeFactorization stages was successively interconnected and the pipeline was executed 100 times for statistical relevance. During execution, the following performance measures were analyzed: (i) the pipeline latency (the time required for one integer to pass through all pipeline stages), (ii) the pipeline throughput (how often an integer is completed per second), (iii) the total execution time for all 100 runs and (iv) the pipeline speedup (the non-threaded execution time divided by the threaded execution time). The benchmark was performed on a desktop computer Performance benchmark using prime factorization equipped with an $Intel^{\mathbb{R}}$ CoreTM i7-860 quad-core processor and the results are shown in Figure F.2. As expected, it can be seen that the pipeline latency increases linearly with the number of stages if the pipeline is executed in non-threaded mode (see Figure F.2a). An identical linear increase could be expected for the threaded mode if a dedicated CPU would be available for each of the *n* stages. However, in our test setup, this is only the case for $n \leq 4$. For higher numbers of stages, the latency progressively increases as more stages contend for CPU time. A similar behavior can be observed for the pipeline throughput. Theoretically, the throughput should decrease inversely proportional to the number of stages for the non-threaded execution mode and remain constant for the threaded mode. The predicted behavior of the non-threaded mode can be well observed in Figure F.2b, whereas the throughput of the threaded mode does not remain constant and substantially drops once the number of stages exceeds the number of physical CPUs. However, a remaining increase in throughput of more than 400 % was observed for n = 20. The last measure that was analyzed is the total execution time for the 100 runs which was also used to compute the overall speed-up. The results of this analysis are shown in Figure F.2c and Figure F.2d, respectively. As expected, the execution time observed for the non-threaded mode is linearly increasing with the number of stages. However, for the threaded mode an about 4 times lower execution time was measured which results in a speed-up close to the theoretical limit of 400 % of the used quad-core CPU.

LIST OF FIGURES

Figure 2.1	Three tion estagorization of tenden driven repets	10
Figure 2.1 Figure 2.2	Three tier categorization of tendon-driven robots Revolving and point-to-point tendon routing .	
Figure 2.2		11
0	Posture independent and dependent lever arms	12
Figure 2.4	Examples of musculoskeletal robots	13
Figure 2.5	Overview of approaches for computing joint	
Figure a (torques from actuator forces	17
Figure 2.6	Examples of analytical modeling approaches for musculoskeletal robots	10
Figure e 1	Rigid body dynamics pipeline.	18
Figure 3.1 Figure 4.1	Muscle dynamics model.	24 20
-		30
Figure 4.2	Model of an electromagnetic actuator.	31
Figure 4.3	Interpolation of gear efficiency for numerical	~ 4
Eigung (stability	34
Figure 4.4	Straight-line muscle kinematics model.	35
Figure 4.5	Example muscle path with via-point	37
Figure 4.6	Spherical wrapping surface example	38
Figure 4.7	Spherical wrapping surface kinematics.	39
Figure 4.8	Spherical wrapping surface state machine.	41
Figure 4.9	Normal <i>flipping</i> and revolution counter incre-	
	ment example.	42
Figure 4.10	Cylindrical wrapping surface model	45
Figure 4.11	Mesh wrapping surface example.	48
Figure 4.12	Comparison of geodesic path algorithms	51
Figure 4.13	Standard linear solid (SLS) or Zener model	53
Figure 4.14	Wiechert model	55
Figure 4.15	Coulomb friction on via-points and spherical	
	wrapping surfaces.	57
Figure 4.16	Proposed data structure of a muscle in a physics-	
	engine.	60
Figure 5.1	Principle of parametric model calibration	66
Figure 5.2	Evolution Strategy.	67
Figure 5.3	Tendon length to joint position ambiguity ex-	
	ample	70
Figure 5.4	Example joint position and muscle force trajec-	
	tories	74
Figure 6.1	Photographs of the developed musculoskeletal	
	robot Anthrob	76
Figure 6.2	Human and Anthrob glenohumeral (shoulder)	
	joint	77
Figure 6.3	Human and Anthrob elbow joint	78
Figure 6.4	Anthrob forearm and hand	79

Figure 6.5	Anthrob glenohumeral and elbow joint muscles. 80
Figure 6.6	Anthrob muscle units 81
Figure 7.1	Anthrob physics-based skeleton model 86
Figure 7.2	Experimental setup of the NBR identification. 88
Figure 7.3	Overview of O-Ring parameters
Figure 7.4	Initial and settled engineering stress vs. strain
	of both recording runs for two NBR O-ring types. 91
Figure 7.5	Example of SLS (or Zener) model fitting for
	ring type #2
Figure 7.6	Example of Wiechert model fitting for ring type
	#2—Part A
Figure 7.7	Example of Wiechert model fitting for ring type
	#2—Part B
Figure 7.8	Summary of the NBR model identification ex-
	periment
Figure 7.9	Identification of kinetic friction coefficient of
	ceramic eyelets
Figure 7.10	Actuator position controller
Figure 7.11	ECU power-circuit identification setup and re-
0	sults
Figure 7.12	Benchmark results of physics-based Anthrob
0 ,	model
Figure 7.13	Glenohumeral joint equilibria
Figure 7.14	Example robot trajectory pair
Figure 7.15	Tier 1 simulation-reality gap of Anthrob gleno-
0, 5	humeral joint model I
Figure 7.16	Tier 1 simulation-reality gap of Anthrob gleno-
0,	humeral joint model II
Figure 7.17	Tier 1 simulation-reality gap of Anthrob gleno-
0 , ,	humeral joint model III
Figure 7.18	Anthrob equilibria and example trajectories 108
Figure 7.19	Tier 2 simulation-reality gap of Anthrob models. 110
Figure 8.1	Anthrob calibration procedure
Figure 8.2	Tier 1 statics calibration results of model I 113
Figure 8.3	Tier 1 statics calibration results of model II 114
Figure 8.4	Tier 1 statics calibration results of model III 115
Figure 8.5	Tier 1 dynamics calibration results of model I. 118
Figure 8.6	Tier 1 dynamics calibration results of model II. 119
Figure 8.7	Tier 1 dynamics calibration results of model III. 120
Figure 8.8	Example trajectory after tier 1 calibration 121
Figure 8.9	Tier 2 statics calibration results
Figure 8.10	Tier 2 dynamics calibration results—part I 126
Figure 8.11	Tier 2 dynamics calibration results—part I 120
Figure 8.12	Example trajectory after tier 2 calibration—part I. 127
Figure 8.13	Example trajectory after tier 2 calibration—part II.129 Example trajectory after tier 2 calibration—part II.129
0	
Figure 8.14	Validation trajectory

Figure 8.15	Validation results—part I	131
Figure 8.16	Validation results—part II	132
Figure 8.17	Anthrob model calibration results summary	134
Figure A.1	Motion capture hardware setup	144
Figure A.2	Image processing pipeline of the developed mo-	
	tion capture system	146
Figure A.3	Marker segmentation	147
Figure A.4	Example of the developed graph-based marker	
	labeling procedure	149
Figure A.5	Image processing pipeline latency	152
Figure B.1	Stress and strain	156
Figure B.2	Maxwell fluid model	157
Figure B.3	Voigt (or Kelvin) solid model	158
Figure C.2	Wrapping meshes of muscle kinematics model III	.165
Figure E.1	CALIPER framework overview	180
Figure E.2	CALIPER plugin management system	181
Figure E.3	Caliper screenshots.	182
Figure F.1	Qt Pipeline Toolkit (QPT) overview	184
Figure F.2	Qt Pipeline Toolkit (QPT) benchmarking results.	185

LIST OF TABLES

Table 2.1	Overview of important musculoskeletal robots	14
Table 4.1	Gear operation modes.	33
Table 4.2	Increment and decrement conditions of the rev-	
	olution counter.	43
Table 6.1	Properties of Anthrob muscles	80
Table 6.2	Anthrob muscle unit types.	82
Table 7.1	Anthrob muscle kinematics models	87
Table 7.2	Tested O-ring types	90
Table 8.1	Overview of calibration object parameters	112
Table A.1	Accuracy of marker segmentation	153
Table C.1	Joint friction parameters of the physics-based	
	Anthrob model	160
Table C.2	Actuator types and parameters	161
Table C.3	Parameters of the Anthrob PD actuator posi-	
	tion controller	161
Table C.4	NBR-70 Zener model fitting results	161
Table C.5	NBR-70 Wiechert model fitting results	161
Table C.6	Parameters of muscle kinematics model I	162
Table C.7	Parameters of muscle kinematics model II	163
Table C.8	Parameters of muscle kinematics model II	164
Table C.9	Parameters of muscle kinematics model III	165
Table C.10	Glenohumeral joint equilibria distances	166
Table C.11	Complete robot equilibria distances	166
Table C.12	Pre-calibration simulation-reality gap summary.	166
Table D.1	Post-calibration simulation-reality gap summary	7.167
Table D.2	Object parameters of Anthrob model I (tier 1,	
	statics).	168
Table D.3	Object parameters of Anthrob model I (tier 1,	
	dynamics)	169
Table D.4	Object parameters of Anthrob model I (tier 2).	170
Table D.5	Object parameters of Anthrob model II (tier 1, statics).	171
Table D.6	Object parameters of Anthrob model II (tier 1,	171
	dynamics).	172
Table D.7	Object parameters of Anthrob model II (tier 2).	173
Table D.8	Object parameters of Anthrob model III (tier 1, statics).	174
Table D.9	Object parameters of Anthrob model III (tier 1,	-/4
	dynamics)	175
Table D.10	Object parameters of Anthrob model III (tier 2).	176
Table D.11	Joint friction object parameters.	177
		11

4.1	Algorithm for the physics-based simulation of muscles.	61
5.1	Evolution strategy algorithm	69

LIST OF ABBREVIATIONS

A/D	Analog/Digital
ANN	Artificial Neural Network
API	Application Programming Interface
BEMF	Back Electromotive Force
CAD	Computer-Aided Design
CAN	Controller Area Network
COLLADA	Collaborative Design Activity
CPU	Central Processing Unit
CSF	Caliper Scene File
DAQ	Data Acquisition
DC	Direct Current
DoF	Degree of Freedom
EA	Evolutionary Algorithms
Eccerobot	Embodied Cognition in a Compliantly Engineered Robot
ECU	Electronic Control Unit
EMG	Electromyography
ES	Evolution Strategy
FIR	Finite Impulse Response
fMRI	Functional Magnetic Resonance Imaging
FPS	Frames Per Second
FSM	Finite State Machine
FSR	Force-Sensitive Resistor
EU	European Union
GUI	Graphical User Interface
IPA	Fraunhofer Institute for Manufacturing Engineering and Automation

ISO	International Organization for Standardization
JSK	Jouhou System Kougaku Laboratory of the University of Tokyo
NBR	Nitrile Butadiene Rubber
PA-2200	Polyamid 2200
PCIe	Peripheral Component Interconnect Express
PILA	Posture-Independent Lever Arm
PDLA	Posture-Dependent Lever Arm
PWM	Pulse-Width Modulation
QPT	Qt Pipeline Toolkit
RMSE	Root Mean Square Error
SEE	Series Elastic Element
SGAL	Scene-Graph Abstraction Layer
SLS	Selective Laser Sintering
SVG	Scalable Vector Graphics
XML	Extensible Markup Language

LIST OF SYMBOLS

Λ	Revolution counter of the spherical and cylindrical wrap- ping surface
η,ρ	Gear forward-mode efficiency, gear-mode dependent effi- ciency
κ,ξ	Scalar coefficients
λ	Offspring population size or expansion ratio
μ	Viscosity of a Newtonian damper or friction coefficient or population size
σ	Stress
ε	Strain
Α	Obstacle or area
В	Rigid body
С	Set of marker centroids
C_H, C_E	Convex hull and convex envelope
D	Set of candidate centroids
Ε	Robot equilibrium or Young's modulus of Hookean spring
E F	Coordinate frame
F	Coordinate frame
F G	Coordinate frame Geodesic path or graph
F G H	Coordinate frame Geodesic path or graph Friction force or half-space
F G H J	Coordinate frame Geodesic path or graph Friction force or half-space Moment of inertia
F G H J K	Coordinate frame Geodesic path or graph Friction force or half-space Moment of inertia Controller gain
F G H J K K _E	Coordinate frame Geodesic path or graph Friction force or half-space Moment of inertia Controller gain BEMF constant
F G H J K K _E K _T	Coordinate frame Geodesic path or graph Friction force or half-space Moment of inertia Controller gain BEMF constant Torque constant
F G H J K K _E K _T L	Coordinate frame Geodesic path or graph Friction force or half-space Moment of inertia Controller gain BEMF constant Torque constant Inductance

Q	Set of equilibria <i>E</i>
R	Resistance
S^x	Wrapping surface
Т	Trajectory
V	Via-point or set of mesh vertices
${\cal F}$	Individual objective function value (fitness)
\mathcal{I}	Individual
$\mathcal{N}(\mu,\sigma)$	Gaussian distribution with mean μ and standard deviation σ
${\cal P}$	Population
$\triangle t$	Simulation time-step
С	Spring damping coefficient
d	Diameter
h	Height
i	Electric current
k	Spring stiffness or viscous friction coefficient
1	Length
т	Mass
r	Radius
t	Time
и	Input or voltage
С	Matrix of Coriolis terms
Ε	Essential matrix
F	Fundamental matrix or matrix of triangulated mesh facets
H	Muscle Jacobian
Ι	Inertia tensor
L	Matrix of lever arms
М	Mass (or inertia) matrix
α, α	Vector of angular and linear accelerations
χ , χ	State of the mesh wrapping surface and of the sphere/cylin- der wrapping surface state machine

ω,ω	Angular velocity vector, angular velocity
τ ,τ	Vector of torques ($\in \mathbb{R}^n$), torque (or relaxation/retardation time)
c, ĉ	Marker centroid ($\in \mathbb{R}^2$ and $\in \mathbb{R}^3$, respectively)
<i>f</i> , <i>f</i>	Force vector ($\in \mathbb{R}^n$), force ($\in \mathbb{R}$)
g , g	Vector of gravity terms, gravity or generation counter
n	Normal vector
q, <i>q</i> , <i>q</i>	Vector of joint positions, velocities and accelerations
S	Vector of strategy parameters
t	Tangent point
v , v	Linear velocity vector, linear velocity
y	Vector of object parameters
Ê	Matrix of muscle lines of action
riangle f	Vector of force errors
riangle q	Vector of angular distances
$^{A}\boldsymbol{R}_{B}$	Rotation matrix of frame <i>B</i> relative to frame <i>A</i>
$^{A}T_{B}$	Transformation matrix of frame B relative to frame A

BIBLIOGRAPHY

- S. Andersson, A. Söderberg, and S. Björklund. Friction models for sliding dry, boundary and mixed lubricated contacts. *Tribology International*, 40(4):580 – 587, 2007.
- [2] Y. Asano, H. Mizoguchi, M. Osada, T. Kozuki, J. Urata, T. Izawa, Y. Nakanishi, K. Okada, and M. Inaba. Biomimetic design of musculoskeletal humanoid knee joint with patella and screw-home mechanism. In *Robotics and Biomimetics* (*ROBIO*), 2011 IEEE International Conference on, pages 1813 –1818, 2011.
- [3] Y. Asano, H. Mizoguchi, T. K. Y. Motegi, M. Osada, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Lower Thigh Design of Detailed Musculoskeletal Humanoid "Kenshiro". In *IEEE/RSJ International Conference on Intelligent Robots* and Systems, 2012.
- [4] K. Athanasiou and R. Natoli. Introduction to Continuum Biomechanics. Synthesis Lectures on Biomedical Engineering Series. Morgan & Claypool, 2008.
- [5] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. ACM Trans. Math. Softw., 22(4):469–483, Dec. 1996.
- [6] H.-G. Beyer and H.-P. Schwefel. Evolution strategies a comprehensive introduction. *Natural Computing*, 1:3–52, May 2002.
- [7] E. Bizzi, N. Accornero, W. Chapple, and N. Hogan. Arm trajectory formation in monkeys. *Exp Brain Res*, 46(1):139–143, 1982.
- [8] A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, pages 281–288, New York, NY, USA, 2007. ACM.
- [9] J. Bouguet. Matlab calibration tool. URL http://www.vision.caltech.edu/ bouguetj/calib_doc/.
- [10] H. F. Brinson and L. C. Brinson. *Polymer Engineering Science and Viscoelasticity: An Introduction.* Springer, 2008.
- [11] F. Bueche. Molecular basis for the mullins effect. *Journal of Applied Polymer Science*, 4(10):107–114, 1960.
- [12] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 49–60, Washington, DC, USA, 1987. IEEE Computer Society.
- [13] J. Chen and Y. Han. Shortest paths on a polyhedron, part i: Computing shortest paths. *International Journal of Computational Geometry & Applications*, 6:127– 144, 1990.
- [14] J. Chestnutt, P. Michel, J. Kuffner, and T. Kanade. Locomotion among dynamic obstacles for the Honda ASIMO. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, pages 2572–2573, Nov 2007.

- [15] J. Chiasson. Modeling and High Performance Control of Electric Machines. Wiley-IEEE Press, 2005.
- [16] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Prentice Hall International, 3rd ed. edition, Sept. 2003.
- [17] R. Crandall and C. Pomerance. Prime Numbers: A Computational Perspective. Lecture notes in statistics. Springer, 2005.
- [18] R. Crawford. Plastics Engineering. Elsevier Science, 3rd edition, 1998.
- [19] V. De Sapio, J. Warren, O. Khatib, and S. Delp. Simulating the task-level control of human motion: a methodology and framework for implementation. *The Visual Computer*, 21:289–302, 2005.
- [20] S. Delp, J. Loan, M. Hoy, F. E. Zajac, E. Topp, and J. Rosen. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *Biomedical Engineering, IEEE Transactions on*, 37(8):757–767, 1990.
- [21] S. Delp, F. Anderson, A. Arnold, P. Loan, A. Habib, C. John, E. Guendelman, and D. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering*, *IEEE Transactions on*, 54(11): 1940–1950, Nov. 2007.
- [22] S. L. Delp and J. P. Loan. A computational framework for simulating and analyzing human and animal movement. *Computing in Science & Engineering*, 2(5):46–55, Oct. 2000.
- [23] E. Desailly, P. Sardain, N. Khouri, D. Yepremian, and P. Lacouture. The convex wrapping algorithm: a method for identifying muscle paths using the underlying bone mesh. J Biomech, 43(13):2601–2607, Sep 2010.
- [24] E. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1(1):269–271, 1959.
- [25] K. Dorfmüller. Robust tracking for augmented reality using retroreflective markers. Computers & Graphics, 23(6):795 – 800, 1999.
- [26] W. F. Dostal and J. G. Andrews. A three-dimensional biomechanical model of hip musculature. *Journal of Biomechanics*, 14(11):803 – 812, 1981.
- [27] D. Eberly. Game Physics. Interactive 3D technology series. Taylor & Francis, 2003.
- [28] A. Engín. On the biomechanics of the shoulder complex. *Journal of Biomechanics*, 13(7):575 – 590, 1980.
- [29] K. Erleben. *Stable, Robust, and Versatile Multibody Dynamics Animation*. PhD thesis, University of Copenhagen, Denmark, November 2004.
- [30] R. Featherstone. *Rigid Body Dynamics Algorithms*. Kluwer international series in engineering and computer science: Robotics. Springer London, Limited, 2008.
- [31] J. Foley. Computer Graphics: Principles and Practice, Second Edition in C. Addison-Wesley systems programming series. Addison-Wesley Pub, 1996.
- [32] R. L. Gajdosik. Passive extensibility of skeletal muscle: review of the literature with clinical implications. *Clinical Biomechanics*, 16(2):87 – 101, 2001.
- [33] D. Gamez, R. Newcombe, O. Holland, and R. Knight. Two simulation tools for biologically inspired virtual robotics. In *Proceedings of the IEEE 5th Chapter Conference on Advances in Cybernetic Systems*, pages 85–90, 2006.

- [34] B. A. Garner and M. G. Pandy. The obstacle-set method for representing muscle paths in musculoskeletal models. *Computer Methods in Biomechanics and Biomedical Engineering*, 3(1):1–30, 2000.
- [35] H. Garstenauer. A Unified Framework for Rigid Body Dynamics. Master's thesis, Johannes Kepler Universität Linz, 2006.
- [36] A. Gaschler. Real-Time Marker-Based Motion Tracking: Application to Kinematic Model Estimation of a Humanoid Robot. Master's thesis, Technische Universität München, Germany, 2011.
- [37] A. Gaschler. Visual motion capturing for kinematic model estimation of a humanoid robot. In *Proceedings of the 33rd international conference on Pattern recognition*, pages 438–443. Springer, 2011.
- [38] H. Gomi and Kawato. Equilibrium-point control hypothesis examined by measured arm stiffness during multijoint movement. *Science*, 272(5258):117–120, Apr 1996.
- [39] L. Gregoire, H. E. Veeger, P. A. Huijing, and G. J. van Ingen Schenau. Role of mono-and biarticular muscles in explosive movements. *Int J Sports Med*, 5(6): 301–305, 1984.
- [40] D. Gross, W. Hauger, J. Schrader, J. Schröder, and W. Wall. *Technische Mechanik* 1: *Statik*. Springer-Lehrbuch. Springer, 2008.
- [41] A. Halim. Human Anatomy: Upper Limb and Thorax. I.K. International Publishing House Pvt., Limited, 2008.
- [42] R. I. Hartley and P. Sturm. Triangulation. Computer Vision and Image Understanding, 68(2):146 – 157, 1997.
- [43] A. V. Hill. The heat of shortening and the dynamic constants of muscle. Proceedings of the Royal Society of London. Series B, Biological Sciences, 126(843):pp. 136–195, 1938.
- [44] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl. DLR's torque-controlled light weight robot III-are we reaching the technological limits now? In *Robotics and Automation*, 2002. Proceedings. ICRA '02. IEEE International Conference on, volume 2, pages 1710 – 1716 vol.2, 2002.
- [45] F. Hoffmeister and T. Bäck. Genetic algorithms and evolution strategies similarities and differences. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, PPSN I, pages 455–469, London, UK, 1991. Springer-Verlag.
- [46] O. Holland and R. Knight. The anthropomimetic principle. In *Proceedings of the AISBo6 Symposium on Biologically Inspired Robotics*, 2006.
- [47] J. M. Hollerbach. Computers, brains and the control of movement. *Trends Neurosci*, 5:189–192, Jan. 1982.
- [48] K. Hongo, Y. Nakanishi, Y. Namiki, I. Mizuuchi, and M. Inaba. Automatic parameter adjustment of reflexive walking of a musculo-skeletal humanoid. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 16–21, dec. 2008.
- [49] K. Hosoda and Y. Ishii. External rotation as morphological bootstrapping for emergence of biped walking. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on,* pages 317–322, aug. 2010.

- [50] H. Imamizu, S. Miyauchi, T. Tamada, Y. Sasaki, R. Takino, B. Pütz, T. Yoshioka, and M. Kawato. Human cerebellar activity reflecting an acquired internal model of a new tool. *Nature*, 403(6766):192–195, Jan 2000.
- [51] N. Ito, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Development of very small high output motor driver for realizing forceful musculoskeletal humanoids. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on,* pages 385–390, dec. 2010.
- [52] T. Izawa, Y. Nakanishi, N. Ito, M. Osada, K. Hongo, S. Ohta, T. Yoshikai, K. Okada, and M. Inaba. Development of stiffness changeable multijoint cervical structure with soft sensor flesh for musculo-skeletal humanoids. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on,* pages 665–670, dec. 2010.
- [53] J. Jackson. Microsoft robotics studio: A technical introduction. Robotics Automation Magazine, IEEE, 14(4):82–87, dec. 2007.
- [54] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers. Design of the Utah/M.I.T. dextrous hand. In *Robotics and Automation. Proceedings.* 1986 IEEE International Conference on, volume 3, pages 1520–1532, Apr. 1986.
- [55] S. C. Jacobsen, H. Ko, E. K. Iversen, and C. C. Davis. Antagonistic control of a tendon driven manipulator. In *Proc. Conf. IEEE Int Robotics and Automation*, pages 1334–1339, 1989.
- [56] S. C. Jacobsen, H. Ko, E. K. Iversen, and C. C. Davis. Control strategies for tendon-driven manipulators. *Control Systems Magazine*, *IEEE*, 10(2):23–28, Feb. 1990.
- [57] S. Jadhav. Advanced Computer Architecture & Computing. Technical Publications, 2009.
- [58] M. Jäntsch. *Non-Linear Control Strategies for Musculoskeletal Robots*. PhD thesis, Technische Universität München, 2013.
- [59] M. Jäntsch, S. Wittmeier, and A. Knoll. Distributed Control for an Anthropomimetic Robot. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 5466 –5471, Oct 2010.
- [60] M. Jäntsch, C. Schmaler, S. Wittmeier, K. Dalamagkidis, and A. Knoll. A scalable Joint-Space Controller for Musculoskeletal Robots with Spherical Joints. In Proc. IEEE International Conference on Robotics and Biomimetics ROBIO 2011, 2011.
- [61] M. Jäntsch, S. Wittmeier, K. Dalamagkidis, and A. Knoll. Computed muscle control for an anthropomimetic elbow joint. In *Intelligent Robots and Systems* (IROS), 2012 IEEE/RSJ International Conference on, pages 2192 –2197, oct. 2012.
- [62] R. Jensen and D. Davy. An investigation of muscle lines of action about the hip: A centroid line approach vs the straight line approach. *Journal of Biomechanics*, 8(2):103 – 110, 1975.
- [63] T. Kanai. Approximate shortest path on a polyhedral surface and its applications. In *Computer-Aided Design*, pages 241–250, 2000.
- [64] E. R. Kandel, J. Schwartz, and T. M. Jessell. Principles of Neural Science. McGraw-Hill, 4th edition, 2000.
- [65] B. Kaneva and J. O'Rourke. An Implementation of Chen & Han's Shortest Paths Algorithm. In Proc. of the 12th Canadian Conference on Computational Geometry, pages 139–146, New Brunswick, August 2000.

- [66] S. Kapoor. Efficient computation of geodesic shortest paths. In In Proc. 32nd Annu. ACM Sympos. Theory Comput, pages 770–779, 1999.
- [67] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
- [68] M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, 1999.
- [69] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.
- [70] S. Klug, T. Lens, O. von Stryk, B. Möhl, and A. Karguth. Biologically inspired robot manipulator for new applications in automation engineering. In *Proc. of Robotik Congress 2008*. VDI Wissensforum GmbH, 2008.
- [71] A. Kochan. Shadow delivers first hand. Industrial Robot: An International Journal, 32(1):15–16, 2005.
- [72] T. Kozuki, H. Mizoguchi, Y. Asano, M. Osada, T. Shirai, U. Junichi, Y. Nakanishi, K. Osada, and M. Inaba. Design Methodology for the Thorax and Shoulder of Human MImetic Musculoskeletal Humanoid Kenshiro—A Thorax structure with Rib like Surface—. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.
- [73] M. Kumamoto, T. Oshima, and T. Yamamoto. Control properties induced by the existence of antagonistic pairs of bi-articular muscles – mechanical engineering model analyses. *Human Movement Science*, 13(5):611–634, Oct. 1994.
- [74] M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proceedings of the thirteenth annual symposium* on Computational geometry, SCG '97, pages 485–486, New York, NY, USA, 1997. ACM.
- [75] S. Lawrence, C. L. Giles, and A. C. Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *In Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, pages 540–545. AAAI Press, 1997.
- [76] J. J. Lee. Tendon-driven manipulators: analysis, synthesis, and control. PhD thesis, University of Maryland, 1991.
- [77] T. Lens, J. Kunz, O. v. Stryk, C. Trommer, and A. Karguth. Biorob-arm: A quickly deployable and intrinsically safe, light- weight robot arm for service robotics applications. *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–6, june 2010.
- [78] A. Maheshwari and S. Wuhrer. Geodesic paths on 3d surfaces: Survey and open problems. *CoRR*, abs/0904.2550, 2009.
- [79] G. Marai, D. Laidlaw, C. Demiralp, S. Andrews, C. Grimm, and J. Crisco. Estimating joint contact areas and ligament lengths from bone kinematics and surfaces. *Biomedical Engineering, IEEE Transactions on*, 51(5):790–799, 2004.
- [80] H. Marques, M. Jäntsch, S. Wittmeier, O. Holland, C. Alessandro, A. Diamond, M. Lungarella, and R. Knight. ECCE1: The first of a series of anthropomimetic musculoskeletal upper torsos. In *10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 391–396, Dec 2010.

- [81] S. P. Marques and G. J. Creus. Computational Viscoelasticity. Springer, 2012.
- [82] O. Masahiko, I. Nobuyuki, N. Yuto, and I. Masayuki. Stiffness readout in musculo-skeletal humanoid robot by using rotary potentiometer. In *Sensors*, 2010 IEEE, pages 2329–2333, 2010.
- [83] N. Matsui and M. Shigyo. Brushless dc motor control without position and speed sensors. *Industry Applications, IEEE Transactions on*, 28(1):120–127, 1992.
- [84] C. Maufroy, H.-M. Maus, K. Radkhah, D. Scholz, O. von Stryk, and A. Seyfarth. Dynamic leg function of the BioBiped humanoid robot. In *Proc. 5th Int. Symposium on Adaptive Motion of Animals and Machines (AMAM)*, Osaka, Japan, Oct. 11-14 2011.
- [85] J. Meriam and L. Kraige. Engineering Mechanics: Statics. Engineering Mechanics: SI Version. Wiley, 2008.
- [86] M. A. Meyers and K. K. Chawla. *Mechanical Behavior of Materials*. Cambridge University Press, 2nd edition, 2009.
- [87] O. Michel. Cyberbotics Ltd. Webots TM : Professional Mobile Robot Simulation. Int. Journal of Advanced Robotic Systems, 1:39–42, 2004.
- [88] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. SIAM J. Comput., 16(4):647–668, Aug. 1987.
- [89] H. Mizoguchi, Y. Asano, T. Izawa, M. Osada, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Biomimetic design and implementation of muscle arrangement around hip joint for musculoskeletal humanoid. In *Robotics and Biomimetics* (*ROBIO*), 2011 IEEE International Conference on, pages 1819–1824, dec. 2011.
- [90] I. Mizuuchi. A musculoskeletal flexible-spine humanoid Kotaro aiming at the future in 15 years' time. *Mobile Robots-Towards New Applications*, pages 45–56, 2006.
- [91] I. Mizuuchi, R. Tajima, T. Yoshikai, D. Sato, K. Nagashima, M. Inaba, Y. Kuniyoshi, and H. Inoue. The design and control of the flexible spine of a fully tendon-driven humanoid "Kenta". In *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, volume 3, pages 2527–2532, 2002.
- [92] I. Mizuuchi, S. Yoshida, T. Yoshikai, M. Inaba, D. Sato, and H. Inoue. Behavior developing environment for the large-dof muscle-driven humanoid equipped with numerous sensors. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1940 – 1945 vol.2, sept. 2003.
- [93] I. Mizuuchi, T. Yoshikai, Y. Nakanishi, Y. Sodeyama, T. Yamamoto, A. Miyadera, T. Niemela, M. Hayashi, J. Urata, and M. Inaba. Development of muscle-driven flexible-spine humanoids. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 339–344, dec. 2005.
- [94] I. Mizuuchi, Y. Nakanishi, Y. Namiki, T. Nishino, J. Urata, M. Inaba, T. Yoshikai, and Y. Sodeyama. Realization of Standing of the Musculoskeletal Humanoid Kotaro by Reinforcing Muscles. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 176–181, 2006.
- [95] I. Mizuuchi, T. Yoshikai, Y. Sodeyama, Y. Nakanishi, A. Miyadera, T. Yamamoto, T. Niemela, M. Hayashi, J. Urata, Y. Namiki, T. Nishino, and M. Inaba. Development of musculoskeletal humanoid Kotaro. In *Robotics and Automation*, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 82–87, May 2006.

- [96] I. Mizuuchi, Y. Nakanishi, Y. Sodeyama, Y. Namiki, T. Nishino, N. Muramatsu, J. Urata, K. Hongo, T. Yoshikai, and M. Inaba. An advanced musculoskeletal humanoid Kojiro. In *Humanoid Robots*, 2007 7th IEEE-RAS International Conference on, pages 294–299, 292007-dec.1 2007.
- [97] R. Murray, Z. Li, S. Sastry, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC PressINC, 1994.
- [98] Y. Nakanishi, I. Mizuuchi, T. Yoshikai, T. Inamura, and M. Inaba. Pedaling by a redundant musculo-skeletal humanoid robot. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 68–73, 2005.
- [99] Y. Nakanishi, Y. Namiki, K. Hongo, J. Urata, I. Mizuuchi, and M. Inaba. Design of the musculoskeletal trunk and realization of powerful motions using spines. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 96– 101, 29 2007-dec. 1 2007.
- [100] Y. Nakanishi, Y. Namiki, K. Hongo, J. Urata, I. Mizuuchi, and M. Inaba. Realization of large joint movement while standing by a musculoskeletal humanoid using its spine and legs coordinately. In *Intelligent Robots and Systems*, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 205–210, sept. 2008.
- [101] Y. Nakanishi, K. Hongo, I. Mizuuchi, and M. Inaba. Joint proprioception acquisition strategy based on joints-muscles topological maps for musculoskeletal humanoids. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1727–1732, may 2010.
- [102] Y. Nakanishi, T. Izawa, M. Osada, N. Ito, S. Ohta, J. Urata, and M. Inaba. Development of musculoskeletal humanoid kenzoh with mechanical compliance changeable tendons by nonlinear spring unit. In *Robotics and Biomimetics (RO-BIO), 2011 IEEE International Conference on*, pages 2384–2389, dec. 2011.
- [103] K. Narioka and K. Hosoda. Motor development of an pneumatic musculoskeletal infant robot. In *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on, pages 963 –968, may 2011.
- [104] K. Narioka, R. Niiyama, Y. Ishii, and K. Hosoda. Pneumatic musculoskeletal infant robots. In Workshop of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.
- [105] R. Niiyama and Y. Kuniyoshi. A pneumatic biped with an artificial musculoskeletal system. In *4th international symposium on adaptive motion of animals and machines (AMAM2008)*, pages 80–81, 2008.
- [106] R. Niiyama and Y. Kuniyoshi. Design principle based on maximum output force profile for a musculoskeletal robot. *Industrial Robot: An International Journal*, 37(3):250–255, 2010.
- [107] R. Niiyama, A. Nagakubo, and Y. Kuniyoshi. Mowgli: A bipedal jumping and landing robot with an artificial musculoskeletal system. In *Robotics and Automation*, 2007 IEEE International Conference on, pages 2546–2551, 2007.
- [108] R. Niiyama, S. Nishikawa, and Y. Kuniyoshi. Athlete robot with applied human muscle activation patterns for bipedal running. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 498 –503, dec. 2010.
- [109] S. Nishikawa, R. Niiyama, and Y. Kuniyoshi. Running motion in a musculoskeletal bipedal robot using muscle activation pattern control based on a human electromyogram. In *The 5th International Symposium on Adaptive Motion* of Animals and Machines (AMAM 2011), pages 15–16, 2011.

- [110] S. Nishikawa, Y. Yamada, K. Shida, and Y. Kuniyoshi. Dynamic motions by a quadruped musculoskeletal robot with angle-dependent moment arms. In *International Workshop on Bio-Inspired Robots*, 2011.
- [111] J. O'Rourke. Computational geometry column 35. Internat. J. Comput. Geom. Appl., 4–5:513–515, 1999. Also in SIGACT News, 30(2):31-32 (1999), Issue 111.
- [112] M. Osada, N. Ito, Y. Nakanishi, and M. Inaba. Stiffness readout in musculoskeletal humanoid robot by using rotary potentiometer. In *Sensors, 2010 IEEE*, pages 2329–2333, nov 2010.
- [113] M. Osada, T. Izawa, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Approach of planar muscle suitable for musculoskeletal humanoids, especially for their body trunk with spine having multiple vertebral. In *Humanoid Robots (Humanoids)*, 2011 11th IEEE-RAS International Conference on, pages 358 –363, oct. 2011.
- [114] C. Pelchen, C. Schweiger, and M. Otter. Modeling and simulating the efficiency of gearboxes and of planetary gearboxes. In *Proceedings of the 2nd International Modelica Conference*, pages 257–266, Oberpfaffenhofen, March 18-19 2002.
- [115] R. Pfeifer, M. Lungarella, and F. Iida. Self-Organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088-1093, Nov. 2007.
- [116] R. Pfeifer, H. G. Marques, and F. Iida. Soft robotics: The next generation of intelligent machines. In F. Rossi, editor, IJCAI. IJCAI/AAAI, 2013.
- [117] P. Pillay and R. Krishnan. Modeling, simulation, and analysis of permanentmagnet motor drives. ii. the brushless dc motor drive. *Industry Applications*, *IEEE Transactions on*, 25(2):274–279, 1989.
- [118] T. Pintaric and H. Kaufmann. Affordable infrared-optical pose-tracking for virtual and augmented reality. In Proc. of Trends and Issues in Tracking for Virtual Environments Workshop, IEEE VR, 2007.
- [119] W. Platzer and G. Spitzer. Color Atlas of Human Anatomy: Locomotor System. Basic Sciences Series. Thieme Medical Publishers, Incorporated, 2009.
- [120] V. Popov. Coulomb's Law of Friction. In Contact Mechanics and Friction, pages 133–154. Springer Berlin Heidelberg, 2010.
- [121] V. Potkonjak, B. Svetozarevic, and O. Holland. Control of Compliant Anthropomimetic Robot Joint. In Symposium on Computational Geometric Methods in Multibody System Dynamics, 2010.
- [122] G. Pratt and M. Williamson. Series elastic actuators. In Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, volume 1, pages 399–406 vol.1, 1995.
- [123] K. Radkhah, C. Maufroy, M. Maus, D. Scholz, A. Seyfarth, and O. von Stryk. Concept and design of the BioBiped1 robot for human-like walking and running. *International Journal of Humanoid Robotics*, 8(3):439–458, 2011.
- [124] K. Radkhah, T. Lens, and O. von Stryk. Detailed Dynamics Modeling of Bio-Biped's Monoarticular and Biarticular Tendon-Driven Actuation System. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pages 4243–4250, 2012.
- [125] I. Rechenberg. Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. PhD thesis, Technische Universität Berlin, 1971.

- [126] J. Reisinger and A. Steininger. The design of a fail-silent processing node for the predictable hard real-time system mars. *Distributed Systems Engineering*, 1 (2):104–111, 1993.
- [127] M. Ribo, A. Pinz, and A. Fuhrmann. A new optical tracking system for virtual and augmented reality applications. In *Instrumentation and Measurement Technology Conference*, 2001. IMTC 2001. Proceedings of the 18th IEEE, volume 3, pages 1932–1936 vol.3, 2001.
- [128] A. Rost and A. Verl. The quadhelix-drive an improved rope actuator for robotic applications. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3254 –3259, may 2010.
- [129] F. Rothling, R. Haschke, J. Steil, and H. Ritter. Platform portable anthropomorphic grasping with the bielefeld 20-dof shadow and 9-dof tum hand. In *Intelligent Robots and Systems*, 2007. IROS 2007. IEEE/RSJ International Conference on, pages 2951–2956, nov 2007.
- [130] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: system overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478 – 2483 vol.3, 2002.
- [131] H. Schiessel, R. Metzler, A. Blumen, and T. F. Nonnenmacher. Generalized viscoelastic models: their fractional equations with solutions. *Journal of Physics* A: Mathematical and General, 28(23):6567, 1995.
- [132] C. Schmaler. Extracting the Muscle Jacobian for Anthropomimetic Robot Control using Machine Learning. Master's thesis, Technische Universität München, 2011.
- [133] D. Scholz, S. Kurowski, K. Radkhah, and O. von Stryk. Bio-inspired motion control of the musculoskeletal BioBiped1 robot based on a learned inverse dynamics model. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 395 –400, oct. 2011.
- [134] M. Schünke, E. Schulte, L. Ross, U. Schumacher, and E. Lamperti. *Thieme Atlas of Anatomy: General Anatomy and Musculoskeletal System*. Number Bd. 433 in Thieme Atlas of Anatomy Series. Thieme Medical Publishers, Incorporated, 2006.
- [135] B. Schwald. *Punktbasiertes 3D-Tracking starrer und dynamischer Modelle mit einem Stereokamerasystem für mixed reality*. PhD thesis, Technische Universiät Darmstadt, 2006.
- [136] H. Schwefel. Collective Phenomena in Evolutionary Systems. In Preprints of the 31st Annual Meeting of the International Society for General System Research, pages 1025–1033, 1987.
- [137] H.-P. Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diplomarbeit, Technische Universität Berlin, 1965.
- [138] N. Schweighofer, M. A. Arbib, and M. Kawato. Role of the cerebellum in reaching movements in humans. i. distributed inverse dynamics control. *Eur J Neurosci*, 10(1):86–94, Jan 1998.
- [139] M. T. Shaw and W. J. MacKnight. *Introduction to Polymer Viscoelasticity*. Wiley, 3rd edition, 2005.

- [140] M. Sherman, A. Seth, and S. Delp. How to compute muscle moment arm using generalized coordinates. Technical report, Stanford University, 2010.
- [141] B. Siciliano and O. Khatib, editors. Handbook of Robotics. Springer, 2008.
- [142] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control.* Springer, 2009.
- [143] Y. Sodeyama, T. Nishino, Y. Namiki, Y. Nakanishi, I. Mizuuchi, and M. Inaba. The designs and motions of a shoulder structure with a spherical thorax, scapulas and collarbones for humanoid Kojiro. In *Intelligent Robots and Systems*, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 1465–1470, sept. 2008.
- [144] J. Sorvari and J. Hämäläinen. Time integration in linear viscoelasticity—a comparative study. *Mechanics of Time-Dependent Materials*, 14(3):307–328, 2010.
- [145] H. Staab, A. Sonnenburg, and C. Hieger. The dohelix-muscle: A novel technical muscle for bionic robots and actuating drive applications. In *Automation Science and Engineering*, 2007. CASE 2007. IEEE International Conference on, pages 306 –311, sept. 2007.
- [146] R. Stribeck. Die wesentlichen Eigenschaften der Gleit- und Rollenlager. Mitteilungen über Forschungsarbeiten auf dem Gebiete des Ingenieurwesens. Heft 7. Julius Springer, 1903.
- [147] I. M. Stuart. Capstan equation for strings with rigidity. British Journal of Applied Physics, 12(10):559, 1961.
- [148] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. ACM Trans. Graph., 24(3):553–560, July 2005.
- [149] S. Suzuki and K. be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985.
- [150] R. Szeliski. Computer Vision: Algorithms and Applications. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
- [151] D. C. Taylor, D. E. Brooks, and J. B. Ryan. Viscoelastic characteristics of muscle: passive stretching versus muscular contractions. *Med Sci Sports Exerc*, 29(12): 1619–1624, Dec 1997.
- [152] B. Tondu and P. Lopez. Modeling and control of McKibben artificial muscle robot actuators. *Control Systems, IEEE*, 20(2):15–38, apr 2000.
- [153] B. Tondu, V. Boitier, and P. Lopez. Naturally compliant robot-arms actuated by McKibben artificial muscles. In Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on, volume 3, pages 2635 – 2640 vol. 3, oct 1994.
- [154] N. Tsagarakis, M. Sinclair, F. Becchi, G. Metta, G. Sandini, and D. Caldwell. Lower body design of the 'iCub' a human-baby like crawling robot. In Proc. IEEE/RAS International Conference on humanoid robots, pages 450–455, dec. 2006.
- [155] P. Tuffield and H. Elias. The shadow robot mimics human actions. *Industrial Robot: An International Journal*, 30(1):56–60, 2003.
- [156] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions* on, 13(4):376–380, 1991.

- [157] J. Urata, Y. Nakanishi, A. Miyadera, I. Mizuuchi, T. Yoshikai, and M. Inaba. A Three-Dimensional Angle Sensor for a Spherical Joint Using a Micro Camera. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4428–4430, 2006.
- [158] G. van den Bergen and D. Gregorius. Game Physics Pearls. A.K. Peters, 2010.
- [159] F. C. T. Van der Helm, H. E. J. Veeger, G. M. Pronk, L. H. V. Van der Woude, and R. H. Rozendal. Geometry parameters for musculoskeletal modelling of the shoulder system. *Journal of Biomechanics*, 25(2):129–144, Feb. 1992.
- [160] G. J. van Ingen Schenau, M. F. Bobbert, and R. H. Rozendal. The unique action of bi-articular muscles in complex movements. *Journal of Anatomy*, 155:1, 1987.
- [161] S. N. Vukosavic. Electrical Machines. Springer, 2013.
- [162] B. Webb. Animals versus animats: Or why not model the real iguana? *Adaptive Behavior*, 17(4):269–286, 2009.
- [163] K. Weicker. Evolutionäre Algorithmen. Teubner, 2007.
- [164] A. Witkin. Physically based modeling: Principles and practice constrained dynamics. In *Computer Graphics*, 2001.
- [165] S. Wittmeier, M. Jäntsch, K. Dalamagkidis, and A. Knoll. Physics-based modeling of an anthropomimetic robot. In *Intelligent Robots and Systems (IROS)*, 2011 IEEE/RSJ International Conference on, pages 4148–4153, 2011.
- [166] S. Wittmeier, M. Jäntsch, K. Dalamagkidis, M. Rickert, H. Marques, and A. Knoll. Caliper: A universal robot simulation framework for tendon-driven robots. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1063–1068, 2011.
- [167] S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntsch, K. Jovanovic, R. Knight, H. G. Marques, P. Milosavljevic, B. Mitra, B. Svetozarevic, V. Potkonjak, R. Pfeifer, A. Knoll, and O. Holland. Toward anthropomimetic robotics: Development, simulation, and control of a musculoskeletal torso. *Artificial Life*, 19(1):171–193, nov 2012.
- [168] S. Wittmeier, A. Gaschler, M. Jäntsch, K. Dalamagkidis, and A. Knoll. Calibration of a physics-based model of an anthropomimetic robot using evolution strategies. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on,* pages 445 –450, oct. 2012.
- [169] S. Wittmeier, M. Jäntsch, K. Dalamagkidis, A. Panos, F. Volkart, and A. Knoll. Anthrob – A Printed Anthropomimetic Robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 342–347, 2013.
- [170] Y. Yamada, S. Nishikawa, K. Shida, and Y. Kuniyoshi. Emergent locomotion patterns from a quadruped pneumatic musculoskeletal robot with spinobulbar model. In *International Workshop on Bio-Inspired Robots*, 2011.
- [171] Y. Yamada, S. Nishikawa, K. Shida, R. Niiyama, and Y. Kuniyoshi. Neuralbody coupling for emergent locomotion: A musculoskeletal quadruped robot with spinobulbar model. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1499–1506, sept. 2011.
- [172] L. Yang and F. Albregtsen. Fast and exact computation of cartesian geometric moments using discrete green's theorem. *Pattern Recognition*, 29(7):1061 – 1073, 1996.

- [173] K. Yoshida, N. Hata, T. Uchida, and Y. Hori. A novel design and realization of robot arm based on the principle of bi-articular muscles. In *Industrial Technol*ogy, 2006. ICIT 2006. IEEE International Conference on, pages 882–886, 2006.
- [174] T. Yoshikai, S. Yoshida, I. Mizuuchi, D. Sato, M. Inaba, and H. Inoue. Multisensor guided behaviors in whole body tendon-driven humanoid kenta. In *Multisensor Fusion and Integration for Intelligent Systems, MFI2003. Proceedings of IEEE International Conference on*, pages 9 – 14, july-1 aug. 2003.
- [175] V. Zatsiorsky. Kinetics of human motion. Human Kinetics Publishers, 2002.
- [176] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis* and Machine Intelligence, IEEE Transactions on, 22(11):1330–1334, 2000.
- [177] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury. Playing it safe: Humanfriendly robots. *Robotics & Automation Magazine*, IEEE, 11(2):12–21, June 2004.