Learning to Control Redundant Musculoskeletal Systems with Neural Networks and SQP: Exploiting Muscle Properties

Danny Driess* Heiko Zimmermann* Simon Wolfen† Dan Suissa† Daniel Haeufle‡
Daniel Hennes* Marc Toussaint* Syn Schmitt†

***Abstract*— Modeling biomechanical musculoskeletal systems reveals that the mapping from muscle stimulations to movement dynamics is highly nonlinear and complex, which makes it difficult to control those systems with classical techniques. In this work, we not only investigate whether machine learning approaches are capable of learning a controller for such systems. We are especially interested in the question if the structure of the musculoskeletal apparatus exhibits properties that are favorable for the learning task. In particular, we consider learning a control policy from target positions to muscle stimulations. To account for the high actuator redundancy of biomechanical systems, our approach uses a learned forward model represented by a neural network and sequential quadratic programming to obtain the control policy, which also enables us to alternate the co-contraction level and hence allows to change the stiffness of the system and to include optimality criteria like small muscle stimulations. Experiments on both a simulated musculoskeletal model of a human arm and a real biomimetic muscle-driven robot show that our approach is able to learn an accurate controller despite high redundancy and nonlinearity, while retaining sample efficiency.**

Video: `https://youtu.be/ODFvqBWOtec`

I. Introduction

The musculoskeletal apparatus of biological organisms, combined with their nervous systems, are able to perform a huge variety of movements with remarkable adaptability.

A central scientific question is to understand how biological systems generate and especially *learn* these movements, both from a physiological and a neural control point of view. Of particular interest is also the influence of the biological structure on control and learnability.

To this end, modeling musculoskeletal systems reveals that muscles incorporate complex contraction dynamics with nonlinear characteristics [1], non-observable states, the mechanical setup is both kinematic and especially high actuation redundant [2] and the nervous system as well as the excitation-contraction dynamics introduces delays [3]. From a technical feedback control perspective, these are challenging and on the first sight even unfavorable system properties, making the design and analysis of controllers for such systems difficult.

In the field of machine learning in robotics, *reinforcement learning* (RL) has shown to be able to learn control policies for high-dimensional, nonlinear and nontrivial dynamical systems [4]. In model-free RL, the dynamical system that should be controlled is usually treated as a black-box, making it a very general approach. However, due to this generality, RL often suffer from two main problems. One is sample efficiency: especially recent approaches based on deep learning [4] require a lot of interaction with the environment, which often limits their applicability to simulated robots. The other problem, which also restricts many approaches to simulation, is safe exploration: taking explorative actions early in the learning process often leads to unstable behavior that could damage the system.

Fig. 1. Real robot model of a human arm consisting of two joints (shoulder and elbow), which are articulated by five pneumatic muscle-spring units (MSUs), allowing movements within the sagittal plane (2D). The two joints are driven separately by two monoarticular MSUs each. Additionally, a fifth MSU acts as a biarticular muscle over both joints, performing shoulder anteversion and elbow flexion.

These considerations lead to our central research questions: Are machine learning methods able to efficiently learn control policies for such complex musculoskeletal systems? Since biological organisms can learn versatile movements despite their complex biomechanical system, does the *morphological structure* exhibit properties that may *simplify* the learning problem in terms of data efficiency or complexity?

To investigate these questions, we propose to combine state of the art biomechanical models, which we realize in simulation and on a real robot, with a learning approach based on neural networks and sequential quadratic programming. The goal is to learn a control policy which produces muscle stimulations leading to desired target positions of the system. Instead of treating the system as just another complex robot for which machine learning methods are applied, our learning approach explicitly exploits the self-stabilizing natural dynamical properties of musculoskeletal systems. Therefore, it is not necessary to learn stability first, which not only simplifies the learning problem and

* Machine Learning and Robotics Lab, University of Stuttgart, Germany.
† Biomechanics and Biorobotics Group, University of Stuttgart, Germany.
‡ Multi-Level Modeling in Motor Control and Rehabilitation Robotics, University of Tübingen, Germany.

This work was funded by the Baden-Württemberg Stiftung in the scope of the NEUROROBOTICS project *DeepControl*.

D. Haeufle was supported by the Ministry of Science, Research and the Arts Baden-Württemberg (Az: 33-7533.-30-20/7/2)

Final version DOI: `https://doi.org/10.1109/ICRA.2018.8463160`

therefore enables us to safely sample from the (real) system at all stages of the learning phase. The characteristics of the musculoskeletal setup also allows to solve the position control task by learning static open-loop muscle stimulations.

However, due to the high redundancy of biomechanical setups, such an inverse controller cannot be learned directly from data. To account for this, a forward model is learned that maps control inputs to equilibrium locations, which is used in an optimization problem to find control inputs that produce desired target positions. This additionally allows to integrate secondary objectives like minimizing muscle stimulations while still reaching the target position and to adjust the stiffness of the system by manipulating the co-contraction level.

Our main contributions are summarized as follows:

- A learning-based approach to control musculoskeletal systems with high actuator redundancy.
- We show that the dynamics induced by the musculoskeletal structure exhibit properties which are favorable for the learning task, if treated properly.
- These properties allow to safely bootstrap the iteratively learned controller to efficiently generate the training data via goal-directed muscle babbling.
- A concept to influence the stiffness of the system by altering the co-contraction via Jacobian nullspaces.

Next, we discuss related work in Sec. II, followed by the details of the musculoskeletal system, realized both in simulation and on a physical robot (Sec. III). The learning framework is presented in Sec. IV. Finally, we provide results to our research question in the experimental Sec. V.

II. Related Work

A. Control of Musculoskeletal Systems

Several works use optimal control techniques to obtain a controller for complex musculoskeletal systems. For example, Liu et al. [5] develop a hierarchical control methodology for a 7 degree of freedom arm model. The authors of [6] control up to 120 muscles for bipedal locomotion. However, these works not only need access to the full musculoskeletal dynamics model, but also to all (hidden) states. These strong assumptions cannot be made to investigate our research questions. Furthermore, compared to the present work, often simplified biomechanical models are used, as also discussed in [6], which neglect activation dynamics, linearize and simplify muscle dynamics [6], [5], do not consider redundant actuation (biarticular muscles) [5] etc.

B. Learning (Inverse) Models in Robotics

Obtaining inverse models plays a central role in robotics. The non-uniqueness problem of learning inverse mappings naturally arises in kinematics. Simple regression methods average over configurations that correspond to the same task space value and therefore could lead to invalid solutions [7]. Multiple approaches have been developed to cope with this. In [8], the authors resolve the non-uniqueness by weighting the samples according to their distance to a homing position. The authors of [9] learn a joint probability distribution of joint angles and end-effector positions to solve the inverse kinematics problem directly on the position level. They describe difficulties in choosing the right kernel/features and especially how the data can be generated. The latter is addressed in [10], where the iteratively learned inverse kinematics model is bootstrapped to generate the relevant data points in the vicinity of already sampled regions. A weighted regression approach similar to [8] is used to learn this function. Due to the efficient data generation methodology, the approach scales up to hyper-redundant manipulators.

Another important instance of inverse models is learning operational space control. Peters et al. [11] also resolve redundancy by weighting. A key insight of their work is to chose the weighting objective in accordance to the structure of torque controlled rigid body dynamics, making the learning possible. Such information is not available a priori for musculoskeletal systems, making the choice of a suitable weighting objective unclear, which limits the applicability of weighting approaches to our problem setting.

C. Tuning Feedback Controllers with Machine Learning

To account for the fact that learning on real systems may lead to damage, one idea is not to learn a controller directly, but to tune the parameters of prestructured feedback controllers. Recently, it has been proposed to use Bayesian optimization as a sample efficient black-box optimization framework for this task [12], [13]. In [12], the focus is on finding parameters for interaction control frameworks that lead to task success in unknown environments with constrained Bayesian optimization, whereas in [13] the focus lies on safe exploration in order to avoid system failure. However, simple prestructured feedback controllers have difficulties in controlling complex musculoskeletal systems.

D. Learning to Control Muscle Driven Robots

Many approaches have been presented to learn controllers for robots driven by pneumatic muscles. Hesselroth et al. [14] propose to use Kohonen self-organizing feature maps to learn pressures for the pneumatic actuators that realize a desired pose of the end-effector. However, each joint is driven by two muscles only, no biarticular ones and no serial elasticity are considered. Furthermore, only one control signal for each muscle pair is specified – the other one is calculated such that both sum up to a constant. Therefore, the dimensionality and redundancy of the control problem is greatly reduced, at the cost of not being able to change the co-contraction and hence the stiffness of the system or no optimality criteria (e.g. small muscle stimulation) can be included without having to relearn. In [15], an adaptive control strategy based on a neural network is proposed for a single pneumatic muscle pair. As in [14], only one control signal is used and no redundancy has to be resolved. The authors of [16] apply local update RL to control a two joint finger driven by four muscles. However, their approach is just suitable to reach one single target position after learning and not a general controller.

To summarize, existing approaches to learn control policies for musculoskeletal systems either include no optimality criterion, consider simpler dynamical models that are not as close to biology as we intend, do not exploit/resolve

the co-contraction or only learn a specific task. To the best of our knowledge, no prior work has addressed the question, whether the nonlinear and complex dynamics of a biomechanical system feature an inherent structure that is favorable for learning.

III. Representation of a Human Arm: Properties of Musculoskeletal Systems

Biological motion systems are articulated by so-called muscle-tendon units (MTUs), which are hierarchical structures consisting of passive and active components with complex nonlinear dynamics. Experiments reveal that MTUs show a characteristic non-linear force-length and force-velocity relation [1], which crucially depend on their activation states (muscle fibers), but are also caused by the purely passive tendons. Since muscles can only produce forces in contraction direction, for each joint at least two muscles in an antagonistic setup are required. Therefore, biological systems are highly actuator redundant [2]. For serial robots, redundancy is usually referred to kinematics, where (infinitely) many joint configurations can lead to the same end-effector pose. In comparison, the redundancy of muscle driven systems additionally refers to a scenario in which (infinitely) many muscle stimulations can lead to the same joint configuration. This redundant, antagonistic actuator setup, together with the characteristic force-length relations, have an important consequence. A vector of constant muscle activations results in a specific static equilibrium state as discussed in Fig. 3. Small perturbations are counteracted by the passive visco-elastic properties of the MTU without the need of any control [17]. Hence, the mechanical system provides a certain static stability in each equilibrium state. Therefore, this natural dynamics is favorable for control [18].

We suggest that a learning algorithm should not only account for this fact, but especially should also exploit these properties. In contrast, many standard RL test tasks only consider stabilization of the system, which would not primarily be necessary for musculoskeletal systems. In this sense, the characteristics of the muscles and their mechanical setup could *simplify* the learning problem *if* treated properly, since then not a major part of the learning has to deal with stability.

A. Simulation Model

The numerical model of a human arm (see Fig. 2) consists of the upper and lower arm, connected with two joints (elbow and shoulder), which are driven by $m = 6$ muscle-tendon units (MTUs). Each joint is actuated by two MTUs separately. Additionally, two biarticular MTUs drive both joints. We use rigid-body dynamics for the bones. The torques $\boldsymbol{\tau} = \mathbf{R}(\mathbf{q})\mathbf{F}_{\text{MTU}}$ applied to the joints are generated through the MTU forces $\mathbf{F}_{\text{MTU}}$ via non-linear moment arms $\mathbf{R}(\mathbf{q})$ that depend on the joint angles $\mathbf{q}$ and muscle path deflection. To calculate the MTU forces itself, we use the model of Haeufle et al. [1], which is a Hill-type model extended with a damping element. This muscle model is

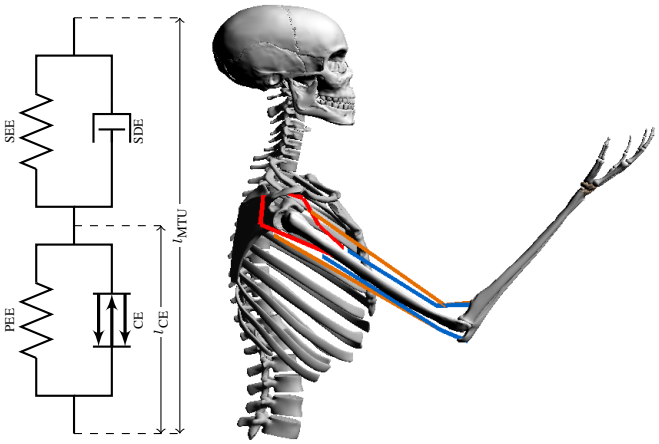


Fig. 2. Right: simulation model of a human arm with elbow and shoulder joint. Red lines depict the two monoarticular shoulder muscles (ante- and retroversion), orange lines the two biarticular ones and blue lines represent the two monoarticular elbow muscles (flexor and extensor). Left: rheological model of each muscle-tendon unit (MTU) with contractile (CE), serial, parallel elastic (SEE, PEE) and damping (SDE) element according to [1].

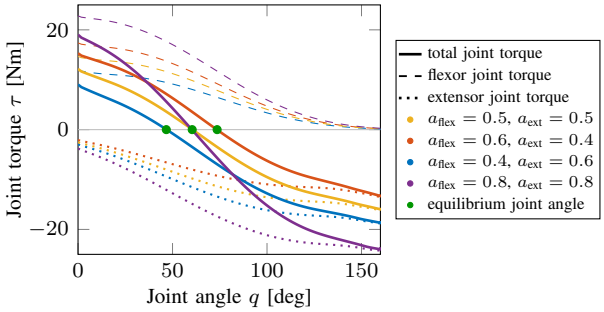


Fig. 3. Relation between static muscle activity a and equilibrium states (green) for an exemplary model of one joint with two muscles. The flexor muscle (dashed lines) generates a positive joint torque, which decreases with q. Its antagonist (extensor) causes a negative torque which increases with q (dotted lines). Those characteristic passive nonlinear force-length relations of muscles depend on their (static) activations a_{flex}, a_{ext} (different colors). The sum of both is the total joint torque τ_{tot} (solid lines). In case of an external perturbation, e.g. increasing the joint angle, those passive properties of the MTUs generate a torque opposing the perturbation (with damping). Therefore, for $\tau_{\text{tot}} = 0$, an equilibrium point (green dots) is reached, whose angle depends on the activations $a_{\text{flex}}, a_{\text{ext}}$ (orange, red, blue line). Further increasing a in both muscles simultaneously (co-contraction) increases the joint stiffness (compare slope at middle green point for orange and violet line), but not the equilibrium angle. Therefore, through the MTU setup, the entire joint behaves like a spring-damper system with tunable rest angle and stiffness/damping by changing activity.

shown in Fig. 2 and follows the nonlinear dynamics

$$\dot{l}_{\text{CE}} = f_{\text{CE}}(l_{\text{MTU}}, \dot{l}_{\text{MTU}}, l_{\text{CE}}, a) \tag{1}$$

$$F_{\text{MTU}} = f_F(l_{\text{MTU}}, \dot{l}_{\text{MTU}}, l_{\text{CE}}, a) \tag{2}$$

with internal state of the length l_{CE} of the contractile element. The transfer from neural muscle stimulations $\mathbf{u} \in [0, 1]^6$, which are the control inputs, to muscle activity follows an additional nonlinear dynamical system

$$\dot{a} = f_a(a, l_{\text{CE}}, u) \tag{3}$$

that introduces delays and is modeled following the Hatze approach [19]. Geometry and muscle parameters are based on [20]. In total, this system with two degrees of freedom has 16 states, of which 12 are unobservable (l_{CE} and a for each muscle), and 6 control inputs. Hand position is $\mathbf{x} \in \mathbb{R}^2$.

B. Bio-Inspired Robot Model of a Human Arm

Our developed physical model is a bio-inspired robot arm (Fig. 1) with two joints that are actuated by five pneumatic muscles of different lengths and thicknesses, representing the respective biological archetype with a nonlinear force-length and force-velocity relation, making them suited for biology-like actuation [21]. For further enhancing the biomimetic characteristics, elastic springs are added in series. That way, these muscle-spring units (MSU) have similar characteristics in their active, passive, static and dynamic behavior as the biological counterpart, the MTU. This means that also on the real robot we can exploit the intrinsic stability of musculoskeletal systems for the learning methodology. The MSU pressure is controlled via digital valves. The stimulation input $\mathbf{u} \in [0,1]^5$ corresponds to pressure values of 0 to 5 bar. Unlike the biological muscle, which is able to contract to 40% and to be stretched up to 170% based on its rest length, the used pneumatic actuators can only contract to 75% of their rest length, limiting the range of motion of our current prototype to 14° in the elbow and 60° in the shoulder joint.

IV. Learning Control Methodology

The goal of the present work is to learn a control policy as the function $\pi : \mathbb{R}^d \to [0,1]^m$, which maps a desired hand position $\mathbf{x}_{\mathrm{ref}} \in \mathbb{R}^d$ to normalized muscles stimulations $\mathbf{u} \in [0,1]^m$ that drive the system to this reference. Due to the structure of the biological system (see Sec. III), each constant muscle stimulation $\mathbf{u}$ shifts the equilibrium to a new position. The attractor properties of this equilibrium, e.g. its stiffness/damping, are dependent on the amount of co-contraction. Therefore, the parameterization as $\pi(\mathbf{x}_{\mathrm{ref}}) = \mathbf{u}$ is sufficient for the reaching control task, no state dependency and no full dynamics model is needed. This allows to sample data from the system of the form $\mathcal{D} = \{(\mathbf{u}_i, \mathbf{x}_i)\}_{i=1}^n$, consisting of muscle stimulations $\mathbf{u}_i$ that have led to the position $\mathbf{x}_i$ of the hand in equilibrium. As discussed in Sec. III, due to the actuator redundancy, learning such an inverse model π directly on the data set, e.g. by minimizing the squared error on a predefined function class, is problematic.

Instead, our approach iteratively learns the forward model $\phi : [0,1]^m \to \mathbb{R}^d$, $\phi(\mathbf{u}) = \mathbf{x}$, from the growing dataset $\mathcal{D}$ during exploration, i.e. a function which predicts the position of the hand in equilibrium for a given muscle signal. In absence of external load, ϕ is an unique mapping, hence directly learnable from data using common function approximators. In our case, ϕ is represented as a feedforward neural network. In order to actually obtain the control policy π from the learned forward mapping ϕ, the idea is to minimize the muscle stimulations under which ϕ predicts the desired position $\mathbf{x}_{\mathrm{ref}}$.

These considerations lead to the nonlinear program

$$\mathbf{u}^* = \pi(\mathbf{x}_{\mathrm{ref}}) = \underset{\mathbf{u}\in\mathbb{R}^m}{\operatorname{argmin}} \|\mathbf{u}\|_{\mathbf{W}}^2 + \lambda \|\mathbf{u} - \mathbf{u}_0\|_2^2 \tag{4a}$$

$$\text{s.t.} \quad \phi(\mathbf{u}) = \mathbf{x}_{\mathrm{ref}} \tag{4b}$$

$$0 \le \mathbf{u} \le 1. \tag{4c}$$

The most important is the equality constraint (4b) to ensure that the desired position is reached. The muscle stimulation $\mathbf{u}$ is (component-wise) bound between 0 and 1 for minimal and full stimulation, respectively. The first term in the objective (4a) represents the goal of minimizing the muscle stimulations (weighted by the positive definite matrix $\mathbf{W}$). The second one in (4a) regularizes the optimization problem such that it stays close to an initial stimulation $\mathbf{u}_0$. Since the optimization is performed based on an *iteratively* learned model ϕ, especially in the beginning of the exploration procedure, the forward model cannot generalize over a large space of desired positions. Without the second term, the optimization problem does not account for this fact. The parameter $\lambda \in \mathbb{R}$ allows to balance the influence of the regularization term.

The forward model and the control policy actually represent three components of the dynamical system: Kinematics, gravity compensation and the role of the co-contraction (including biarticular muscles). The muscles itself act as a stabilizing (physical feedback) controller. This indicates that the learning problem is simplified through the biological structure, in the sense that morphology re-represents the control problem to another one, which can be learned more efficiently. On the other hand, the learning algorithm has to cope with high actuator redundancy.

A. Optimization via Sequential Quadratic Programming

To evaluate the control policy π for a desired $\mathbf{x}_{\mathrm{ref}}$, the optimization problem (4) has to be solved. The quadratic objective (4a) is convex, as well as the box constraint (4c). Unfortunately, the most important part, the constraint (4b), is, in general, non-convex. However, since we are mainly interested in finding a muscle stimulation $\mathbf{u}$ that fulfills the constraint (4b) and is close to an initial stimulation $\mathbf{u}_0$ (as discussed above), converging to a local optimum is sufficient. Therefore, sequential quadratic programming is well suited for this optimization problem. We use the algorithm from [22]. The initial starting point for the optimization algorithm is the same $\mathbf{u}_0$ used for the regularization specified above.

B. Jacobian w.r.t. Muscle Stimulations

Given the learned forward model ϕ, its derivative with respect to the muscle stimulations

$$\mathbf{J}(\mathbf{u}) = \frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}) \in \mathbb{R}^{d\times m} \tag{5}$$

locally describes how the position of the system changes for a small change $\delta\mathbf{u}$ in the muscle stimulation as a linear dependency $\delta\mathbf{x} = \mathbf{J}(\mathbf{u})\delta\mathbf{u}$.

Utilizing this Jacobian, the optimization problem can further be analyzed. Assume that the inequality constraints are not active and that $\mathbf{J}(\mathbf{u})$ has full row rank. After linearization of the constraint (4b) around $\tilde{\mathbf{u}}$, the quadratic program

$$\min_{\mathbf{u}\in\mathbb{R}^m} \|\mathbf{u}\|_{\mathbf{W}}^2 + \lambda \|\mathbf{u} - \mathbf{u}_0\|_2^2 \tag{6a}$$

$$\text{s.t.} \quad \phi(\tilde{\mathbf{u}}) + \mathbf{J}(\tilde{\mathbf{u}})(\mathbf{u} - \tilde{\mathbf{u}}) = \mathbf{x}_{\mathrm{ref}} \tag{6b}$$

can be solved analytically

$$\mathbf{u}^* = \tilde{\mathbf{u}} + \mathbf{P}_{\mathbf{W},\lambda}(\tilde{\mathbf{u}})(\mathbf{x}_{\mathrm{ref}} - \phi(\tilde{\mathbf{u}})) + (\mathbf{I} - \mathbf{P}_{\mathbf{W},\lambda}(\tilde{\mathbf{u}})\mathbf{J}(\tilde{\mathbf{u}}))\left(\mathbf{W}_\lambda^{-1}\lambda\mathbf{u}_0 - \tilde{\mathbf{u}}\right) \tag{7}$$

with $\mathbf{P}_{\mathbf{W},\lambda}(\tilde{\mathbf{u}}) = \mathbf{W}_\lambda^{-1}\mathbf{J}(\tilde{\mathbf{u}})^T \left(\mathbf{J}(\tilde{\mathbf{u}})\mathbf{W}_\lambda^{-1}\mathbf{J}(\tilde{\mathbf{u}})^T\right)^{-1}$ and $\mathbf{W}_\lambda = \mathbf{W} + \lambda\mathbf{I}$.

1) Feedback: If we linearize ϕ at the current *measured* position $\tilde{\mathbf{x}}$ for the current stimulation $\tilde{\mathbf{u}}$, then (7) can be used to close the loop. For example, by setting $\mathbf{u}_0 = \tilde{\mathbf{u}}$, $\mathbf{W} = 0$ and introducing a stepsize $\alpha < 1$, we obtain

$$\mathbf{u}_{\text{closed}} = \tilde{\mathbf{u}} + \alpha \cdot \mathbf{J}(\tilde{\mathbf{u}})^T \left(\mathbf{J}(\tilde{\mathbf{u}})\mathbf{J}(\tilde{\mathbf{u}})^T\right)^{-1} (\mathbf{x}_{\text{ref}} - \tilde{\mathbf{x}}), \quad (8)$$

which in the kinematics case is known as Jacobian pseudoinverse motion rate control. This closed-loop controller based on the muscle Jacobian (5) can be combined with the static policy π obtained from the optimization problem. If the goal is to reach a certain target position $\mathbf{x}_{\text{ref}}$, first $\mathbf{u}_{\text{static}} = \pi(\mathbf{x}_{\text{ref}})$ calculates a static muscle stimulation for the majority of the movement, which is executed in a self-stabilizing manner through the musculoskeletal system. In the vicinity of the goal, the feedback can be used to generate fine movements.

2) Influencing the Stiffness of the System by Exploiting the Nullspace of the Muscle Jacobian: Adjusting end-effector stiffness is important for the interaction with the environment [23]. In our setup, the nullspace of the forward model ϕ

$$N_{\text{muscle}}(\mathbf{x}_{\text{ref}}) = \{\, \mathbf{u} \in [0,1]^m \,|\, \phi(\mathbf{u}) = \mathbf{x}_{\text{ref}} \,\} \quad (9)$$

is the set of all *muscle stimulations* that lead to the same position $\mathbf{x}_{\text{ref}}$ in equilibrium. Again using linearization around $\tilde{\mathbf{u}}$, this muscle nullspace can locally be obtained through the nullspace projector of the muscle Jacobian (5). Projecting small perturbations $\delta\mathbf{a} \in \mathbb{R}^m$ into the nullspace of this Jacobian, changes in the muscle stimulation of

$$\delta\mathbf{u} = \left(\mathbf{I} - \mathbf{J}(\tilde{\mathbf{u}})^T \left(\mathbf{J}(\tilde{\mathbf{u}})\mathbf{J}(\tilde{\mathbf{u}})^T\right)^{-1} \mathbf{J}(\tilde{\mathbf{u}})\right) \delta\mathbf{a} \quad (10)$$

do not alter the position of the system, but the *co-contraction*. Therefore, in contrast to kinematics where the nullspace represents the set of all joint configurations corresponding to the same task value, this muscle Jacobian nullspace (9) allows to influence the *stiffness* of the biological system, due to the nonlinear force-length relations of MTUs, via different co-contraction levels (cf. Sec. III). If, for example, $\delta\mathbf{a}$ contains a positive entry for one muscle, then the nullspace projector in (10) has to increase the stimulation for other muscles as well to ensure that the position does not change. This resulting static $\mathbf{u}$ changes the stiffness based on the passive properties of the muscles for a static activation, no control is needed.

C. Data Generation via Goal-Directed Muscle Babbling

One key problem in learning the forward model ϕ is the data generation. In our case, the 6D space of muscle stimulations makes systematic sampling impossible. To overcome this, we adapt the idea of Rolf et al. [10], who proposed to *bootstrap* iteratively learned models to efficiently generate relevant data. Starting with an initial $\mathbf{u}$ leading to $\mathbf{x}$, the model ϕ is trained on this single data point. Then the optimization problem (4) based on this currently learned ϕ is used to predict the stimulation for a new goal position, which is close to the previous one, since in the vicinity of already sampled data points it is expected that the model is not too far off. The outcome is stored in the dataset, ϕ is trained again and procedure is repeated. This goal-directed sampling enables to efficiently explore the relevant stimulation space.

In this context, the importance of the regularizing term in the objective (4a) is emphasized, since the data generation process relies on staying close to the samples in the dataset.

D. Exploration Noise

Especially in the beginning of the learning procedure, exploration in the space of muscle stimulations is necessary, otherwise, no solution for a specific region of the workspace can be found at all or no good one in terms of secondary objectives. Using the control policy to generate the data as described in the last paragraph already introduces exploration, since the forward model is not perfect in the beginning. However, we found empirically that randomness to some extend is favorable. On the other hand, too much exploration noise would counteract the advantages of goal-directed muscle babbling. Therefore, we add Gaussian noise to the muscle stimulations

$$\mathbf{u}_{\text{noise}} = \beta(n) \cdot \mathcal{N}(0, \mathbf{\Sigma}) \in \mathbb{R}^m, \quad (11)$$

centered around zero, multiplied with the scalar term $\beta(n)$ which decreases the exploration noise with the number of sampled data points n. The covariance of the Gaussian distribution is a block diagonal matrix $\mathbf{\Sigma} = \text{diag}\,(\mathbf{C}, \ldots, \mathbf{C}) \in \mathbb{R}^{m\times m}$ with blocks

$$\mathbf{C} = \mathbf{V} \begin{pmatrix} \sigma_1^2 & \\ & \sigma_2^2 \end{pmatrix} \mathbf{V}^{-1} \in \mathbb{R}^{2\times 2}, \quad \mathbf{V} = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \quad (12)$$

acting on each pair of antagonistic muscles. The eigenvectors of $\mathbf{C}$ are chosen such that for $\sigma_1 > \sigma_2$ it is more likely to sample stimulation pairs where one muscle is activated more while its antagonist less (and the other way round).

E. Other Methods for Learning Redundant Inverse Models

In this paragraph, we show how two other approaches for learning inverse models can be applied to our problem setting. In Sec. V, those are evaluated against our approach.

1) Learning with a Distal Teacher: The distal teacher framework [7] resolves the non-uniqueness problem in learning inverse models by obtaining an identity mapping across the concatenation of a learned inverse and forward model. The unique forward mapping ϕ is iteratively trained to predict the hand position $\mathbf{x}$ for a given static muscle stimulation $\mathbf{u}$ as in our method (see Sec. IV). The actual inverse model π is then trained to approximate the identity mapping $\phi(\pi(\mathbf{x})) = \mathbf{x}$ on the dataset, while keeping the forward model fixed. In our case where the forward model is a many-to-one mapping, this approach is able to resolve the ambiguity and learns one particular inverse model. However, which of the infinitely many inverse models is learned can hardly be controlled. Both π and ϕ are neural networks here. To account for $\mathbf{u} \in [0,1]^m$, π has a sigmoid output layer.

2) Direct Learning of the Inverse via Weighting: As briefly described in Sec. II, one way to cope with the non-uniqueness of inverse models is to *weight* the samples according to an objective. This has been used for learning inverse kinematics and operational space control [8], [10],

[11]. To adapt this idea to our setting, it is necessary to define a reasonable weighting objective, which is not clear for musculoskeletal systems. In this case, we train the inverse model π, which is a neural network with a sigmoid output layer to account for $\mathbf{u} \in [0,1]^m$, to minimize the loss

$$\sum_{i=1}^{n} \exp\left(-\|\mathbf{u}_i\|_2^2\right) \|\pi(\mathbf{x}_i) - \mathbf{u}_i\|_2^2, \tag{13}$$

which is a standard squared error weighted by the muscle stimulations $\mathbf{u}_i$ from the dataset. If $\mathbf{u}_i$ has a high magnitude, then this sample is weighted down. This way, the neural network is biased towards small muscle simulations, especially for redundant samples. Since this still lead to (weighted) averaging, the performance is expected to be worse compared to the hard constraint in our optimization problem. Furthermore, there are certain targets $\mathbf{x}$ for which muscle stimulations intrinsically need to be higher than for others. Those also get weighted down, which is unfavorable.

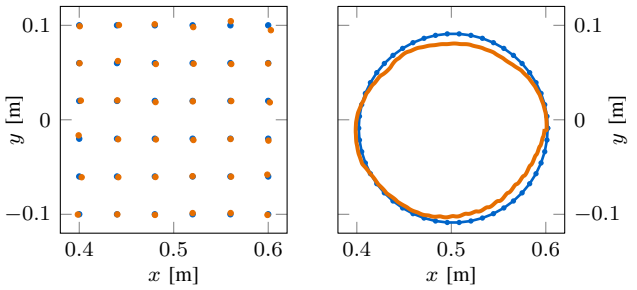
V. Experiments

We demonstrate our approach both in simulation and on the real robot for the task of reaching desired hand positions. In simulation, we compare our methodology to related work that deals with learning inverse models. Furthermore, we show how the stiffness of the system can be changed using our approach. Please also refer to the video attachment.

The learning procedure for all experiments is performed as follows. An initial muscle stimulation $\mathbf{u}_0$ is specified manually such that the hand of the arm is located in the middle of a desired 2D workspace. This $\mathbf{u}_0$ is also used for the regularization in (4a). Each training execution starts from the same rest position where the muscles have zero stimulation. After a data point $(\mathbf{u}, \mathbf{x})$ consisting of the muscle stimulation $\mathbf{u}$ leading to the position $\mathbf{x}$ in equilibrium is sampled, the neural network is trained with stochastic gradient descent for 20 epochs with a batch-size of 32 on the whole dataset, initialized with the weights of the previous iteration. Then, the next target position is chosen according to section IV-C close to the last target (1 cm away). Afterwards, this desired target is fed into the optimization problem (4) which is solved quickly with SQP to estimate the next $\mathbf{u}$. After returning to the rest configuration with $\mathbf{u} = 0$, the procedure starts again. The neural network architecture for all experiments is build from two hidden layers with 300 units each and relu activation functions. To test the generalization capabilities, we evaluate the performance by reaching motions on a grid, whose points have not been part of the training.

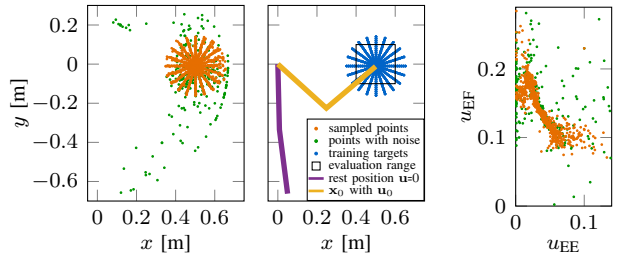
A. Simulation

The initial muscle stimulation was set to $\mathbf{u}_0 = (0.23, 0.1, 0.1, 0.1, 0.2, 0.1)$, corresponding to elbow flexor, extensor (EF, EE), biarticular flexor, extensor (BiF, BiE) and shoulder muscles (SF, SE). As visualized in Fig. 5a, we sampled 1200 data points in four batches with the same 300 target positions with center at $\mathbf{x}_0$ of $\mathbf{u}_0$. The noise scheduling $\beta(n)$ was chosen such that for the first 100 iterations $\beta(n) = 1$, afterwards it decreases exponentially until $n = 300$, where $\beta(n) \approx 0$, such that 300 data points include noise, the other 900 were sampled only using the model. The noise parameters were $\sigma_1 = \frac{1}{15}$, $\sigma_2 = \frac{1}{30}$. The regularization parameter in (4a) was $\lambda = 1$. In Fig. 5b, the stimulations of the elbow flexor and extensor in the dataset are plotted against each other. One can see that the learning algorithm has revealed the anti-correlated antagonistic property, which is favorable for energy efficiency.



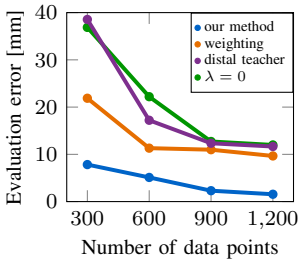
(a) Point reaching evaluation. (b) Circle trajectory following.

Fig. 4. Simulated experiments: Evaluation of accuracy for point reaching movements (a), error 1.6 ± 1.1 mm. Reaching order randomly selected, started from the previous point. Blue is the reference, orange the measured value. (b) Trajectory following accuracy 8 mm for 5 s execution time. Blue dots in (b) denote the 50 target points of the reference.



(a) Sampled data points, training targets, evaluation range as well as arm rest and $\mathbf{x}_0$ configuration. (b) Elbow flex. and ext. stimulation.

Fig. 5. Visualization of data point sampling and arm configuration.



(a) Evolution of evaluation error.

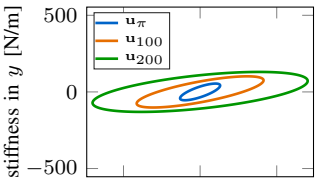
Method	Mean stimulation	Error [mm]
Our approach	**0.082**	**1.6 ± 1.1**
Weighting	0.154	9.7 ± 13.4
Distal teacher	0.308	11.7 ± 6.3
Random + SQP	0.083	87.8 ± 48.3
$\lambda = 0$	0.055	12.0 ± 9.7

(b) Comparison after 1200 data points.

Fig. 6. Comparison of different control policy learning methods for the simulated point reaching experiment. Our method outperformed related approaches both in terms of accuracy and small muscle stimulation. All relevant hyperparameters were the same.

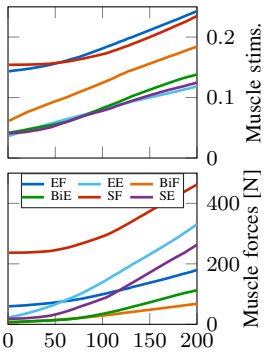
Fig. 4a shows the accuracy of the learned control policy with our proposed approach after a total of 1200 sampled data points for the task of reaching desired locations. The reaching order was randomly selected and started from the previous point after the equilibrium had been reached. The averaged error was 1.6 ± 1.1 mm. Although only trained for point reaching motions, the learned control policy is also capable of following slow trajectories, as can be seen in Fig. 4b, in which a circle was followed in 5 s with an error of 8 mm. 50 target points have been used for the circle reference.

In Fig. 6, we compare our approach to related ones mentioned in IV-E. Both in terms of accuracy and minimal muscle stimulation, our approach outperforms the others. The



(a) Stiffness ellipsoids.

Fig. 7. Stiffness modulation of hand. (a) measured stiffness ellipsoids at $\mathbf{x} = (0.5, 0.0)$ for $\mathbf{u}_\pi = \pi(\mathbf{x})$ and $\mathbf{u}_{100}$ for 100, $\mathbf{u}_{200}$ for 200 nullspace projections. (b) increasing muscle stimulation over the number of projection and the induced muscle forces.



(b) Muscle stimulations $\mathbf{u}$ (top) and forces (bottom).

weighted regression method used twice as much stimulations in average and was about 6 times less accurate than our approach. The distal teacher required about four times higher muscle activations. The error of our method after only 300 data points was even lower than for the others after 1200.

1) Importance of Goal-Directed Muscle Babbling: To support our claim that bootstrapping the incrementally learned control policy to generate the data is essential, we uniformly sampled the 6D $\mathbf{u}$ space and applied those stimulations on the system (1200 data points as before). Based on this dataset, we trained the same forward model ϕ and used the optimization problem (4) to obtain the policy. With an evaluation error of 87.8 $\pm$ 48.3mm (Fig. 6) in the end, it turned out that 1200 random points in a 6D space are not enough to learn generalizable models for this problem.

2) Influence of λ: As can be seen in Fig. 6, the performance of our approach without the regularization, i.e. $\lambda = 0$ in (4a), is, with an evaluation error of 12 mm, much worse than with $\lambda = 1$ (error 1.6 mm) and in the range of the other methods. This can be explained by the fact that 1200 points in a 6D space are very sparse, but the optimization problem treats ϕ as perfect everywhere in the $\mathbf{u}$ space. Therefore, the error is higher for $\lambda = 0$. By introducing the regularization term with $\lambda > 0$, this is prevented since then the optimizer searches in regions close to the data samples. For example, out of the 36 evaluation points, 30 points had at least one zero value in a stimulation component for $\lambda = 0$, which is physiologically unreasonable. $\lambda = 0$ also yielded the lowest mean stimulation. In comparison, for $\lambda = 1$, not a single one showed zero stimulations. In the training dataset, from the 1200 points, 910 for $\lambda = 0$ showed this behavior, whereas only 133 for $\lambda = 1$.

3) Stiffness Adjustment via Muscle Jacobian Nullspace: As developed in Sec. IV-B.2, the nullspace of the Jacobian (5) of the learned model ϕ can be used to increase the co-contraction and hence the stiffness without changing the position. Since neither the linearization nor the learned Jacobian are perfect, we use our proposed local feedback (sec. IV-B.1) to stay at the same position more precisely, i.e. after being initialized with $\mathbf{u} = \pi(\mathbf{x}_{\text{ref}})$, $\mathbf{u}$ is updated with

$$\mathbf{u} \leftarrow \mathbf{u} + 0.5 \cdot \mathbf{J}(\mathbf{u})^T \left(\mathbf{J}(\mathbf{u})\mathbf{J}(\mathbf{u})^T\right)^{-1} (\mathbf{x}_{\text{ref}} - \mathbf{x}) + 0.01 \cdot \left(\mathbf{I} - \mathbf{J}(\mathbf{u})^T \left(\mathbf{J}(\mathbf{u})\mathbf{J}(\mathbf{u})^T\right)^{-1} \mathbf{J}(\mathbf{u})\right) \delta\mathbf{a}, \quad (14)$$

where $\mathbf{x}$ is the current position after each update. Fig. 7b shows increasing muscle stimulations and forces for $\delta\mathbf{a} = \big(0.1, 0.0, 0.1, 0.0, 0.1, 0.0\big)$ and $\mathbf{x}_{\text{ref}} = (0.5, 0)$, while the hand has moved only 1 mm in x and 2 mm in y. Without the feedback, it moved 8 mm in x and 20 mm in y. This increased co-contraction influences the stiffness ellipsoid at $\mathbf{x}_{\text{ref}}$, which was measured and is visualized in Fig. 7a. This stiffness purely relies on the properties of the muscles for static stimulation and therefore requires no external control.

B. Real Robot

For the real robot, an initial muscle stimulation (normalized pressures applied to the pneumatic actuators), of $\mathbf{u}_0 = \left(\frac{1}{5}, \frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$ was specified. Full muscle stimulation of one corresponds to a pressure of 5 bar. The 700 collected data points in total are visualized in Fig. 8a. 200 of them included exploration noise (without any scheduling) with noise parameters $\sigma_1 = 0.3$, $\sigma_2 = 0.14$. As mentioned in Sec. III-B, the elbow joint of our current robot prototype has a range of motion of only $14°$. Therefore, although small, the evaluation covers the major range of motion also in x-direction. Fig. 8b shows the precision of the controller to reach the desired target positions during training, after each batch of 100 data points (excluding the 200 samples generated with noise). Evaluating the performance after training on a grid of 32 target points not part of the training data yielded an error of 8 $\pm$ 4 mm and is visualized in Fig. 8d. Note that due to hysteresis effects of the pneumatic muscles, the reaching motion always started from the rest position with $\mathbf{u} = 0$, as in training. When we tuned muscle stimulations by hand, we hardly used the biarticular muscle, since then the joint angles can be adjusted independently. Interestingly, as can be seen in Fig. 8c, the learned control policy assigns the highest pressure in average to this biarticular muscle. A possible explanation is that the middle of the considered workspace can be reached energy efficiently by using the biarticular muscle alone, which our methodology could discover.

VI. Conclusion

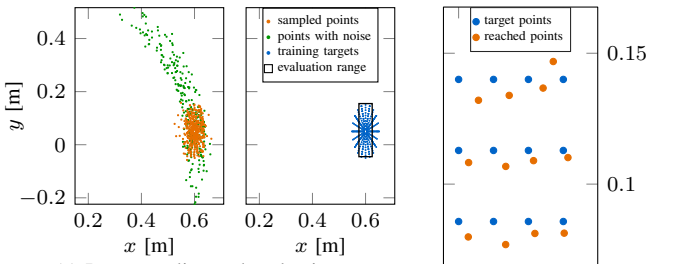
In the present work, we presented a methodology to learn control policies for musculoskeletal systems, which explicitly exploits the benefits of muscle-driven systems.

Compared to existing methods, our approach allows to use the actuation redundancy to modulate the stiffness of the system at a desired position by different co-contraction levels and to pursue additional objectives like small muscle stimulations.

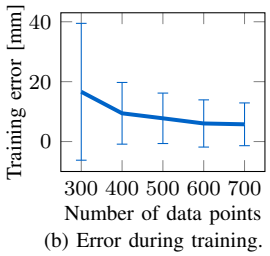
It was important to include a regularizing term to prevent the optimization algorithm to search for physiologically unreasonable solutions at the boundary of the feasible set or in unexplored parameter regions, especially if the dataset consists of few samples only.

Without bootstrapping the iteratively learned model, relevant data could not have been generated efficiently in the high dimensional stimulation space.

The complex biological structure indeed simplifies the learning problem, since the dynamical properties of the muscles and their redundant antagonistic setup provides a



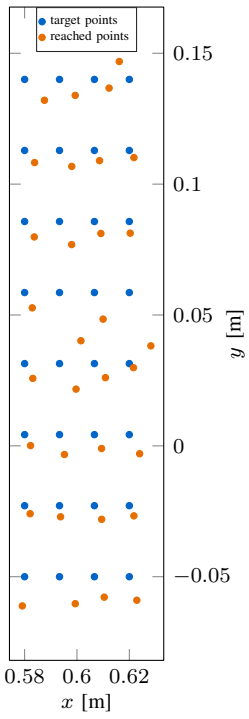
(a) Data sampling and evaluation range.



(b) Error during training.

Muscle	Mean stim.
Shoulder Flex.	1.1 ± 0.5 bar
Shoulder Ext.	1.1 ± 0.5 bar
Biarticular	1.3 ± 0.6 bar
Elbow Flex.	0.8 ± 0.7 bar
Elbow Ext.	0.6 ± 0.6 bar

(c) Average muscle stimulations of training dataset.



(d) Point reaching evaluation, error 8 ± 4 mm.

Fig. 8. Real robot experiments. (a) shows the target star for training, the actual sampled data points (700 in total, 200 with noise) and the evaluation range (black rectangle) for (d). (b) shows the mean reaching precision during training. (c) shows the mean muscle stimulations. (d) shows the point reaching evaluation after training for 32 targets not part of the training set. Each point is reached starting from the rest position with zero stimulation.

certain intrinsic stability to the system. Therefore, a learning algorithm does not necessarily have to learn a stabilizing feedback loop. This induces that first an open-loop control policy that generates a static muscle stimulation for a desired position is sufficient to realize our goal of reaching control. This greatly reduces the dimensionality of the learning problem, because no complete dynamic model as for a torque controlled robot has to be learned. Secondly, the iteratively learned model can safely be bootstrapped to generate relevant training data efficiently, even in the beginning of the learning phase where the model is inaccurate, which would also not directly be possible with a torque driven robot without an already available controller or the risk of damage.

Nevertheless, due to the high redundancy, learning such a mapping is still a nontrivial problem and existing methods known from the inverse kinematics literature were empirically shown to perform inferior, not only in terms of accuracy, but also with respect to minimal muscle stimulations. Furthermore, existing methods also have not considered how the stiffness of a musculoskeletal system can be modulated with a learning approach.

The main limitation of our approach on the real robot is that due to friction, hysteresis and temperature effects, the forward mapping ϕ is not state- and history-independent (in contrast to the simulation), which means that the control accuracy on the real robot could only be reached when starting the movement from the rest position as in training.

We belief that with this work we showed that it is worth to investigate the interaction between the properties of the physical structure and the learning algorithm.

References

[1] D. Haeufle, M. Günther, A. Bayer, and S. Schmitt, "Hill-type muscle model with serial damping and eccentric force-velocity relation." *Journal of Biomechanics*, vol. 47, no. 6, 2014.

[2] R. Shadmehr, *Actuator and Kinematic Redundancy in Biological Motor Control*. Springer Berlin Heidelberg, 1991.

[3] F. Mörl, T. Siebert, S. Schmitt, R. Blickhan, and M. Günther, "Electro-mechanical delay in Hill-type muscle models," *Journal of Mechanics in Medicine and Biology*, vol. 12, no. 5, 2012.

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

[5] D. Liu and E. Todorov, "Hierarchical optimal control of a 7-dof arm model," in *Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009.

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

[7] M. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science*, 1992.

[8] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[9] B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schoelkopf, and J. Peters, "Learning inverse kinematics with structured prediction," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[10] M. Rolf, J. J. Steil, and M. Gienger, "Online Goal Babbling for rapid bootstrapping of inverse models in high dimensions," in *Proc. of the Int. Conf. on Development, Learning and Epigenetic Robotics*, 2011.

[11] J. Peters and S. Schaal, "Learning to control in operational space," *Int. Journal of Robotics Research*, 2008.

[12] D. Drieß, P. Englert, and M. Toussaint, "Constrained bayesian optimization of combined interaction force/task space controllers for manipulations," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[13] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[14] T. Hesselroth, K. Sarkar, P. P. Van Der Smagt, and K. Schulten, "Neural network control of a pneumatic robot arm," *IEEE Transactions on Systems, Man, and Cybernetics*, 1994.

[15] P. Van der Smagt and K. Schulten, "Control of pneumatic robot arm dynamics by a neural network," in *Proc. of the World Congress on Neural Networks*, 1993.

[16] Y. Cui, T. Matsubara, and K. Sugimoto, "Pneumatic artificial muscle-driven robot control using local update reinforcement learning," *Advanced Robotics*, 2017.

[17] I. Brown, S. Scott, and G. Loeb, "Preflexes — programmable high-gain zero-delay intrinsic responses of perturbed musculoskeletal systems," *Society of Neuroscience, Abstracts*, vol. 21, 1995.

[18] D. F. B. Haeufle, M. Günther, G. Wunner, and S. Schmitt, "Quantifying control effort of biological and technical movements: An information-entropy-based approach," *Physical Review E*, vol. 89, no. 1, 2014.

[19] H. Hatze, "A general myocybernetic control model of skeletal muscle," *Biological Cybernetics*, vol. 28, no. 3, pp. 143–157, 1978.

[20] A. Bayer, S. Schmitt, M. Günther, and D. Haeufle, "The influence of biophysical muscle properties on simulating fast human arm movements," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 20, no. 8, 2017.

[21] G. Klute, J. Czerniecki, and B. Hannaford, "McKibben artificial muscles: pneumatic actuators with biomechanical intelligence," in *in Proc. of the Int. Conf. on Advanced Intelligent Mechatronics*, 1999.

[22] D. Kraft, *A software package for sequential quadratic programming*. DFVLR, 1988.

[23] D. Driess, P. Englert, and M. Toussaint, "Active learning with query paths for tactile object shape exploration," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.