Robot Learning for Muscular Systems

Büchler, Dieter

(2019)

Robot Learning for Muscular Systems

Lernen auf Muskelbasierten Robotern
Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation von M.Sc. Dieter Büchler aus Duschanbe
Dezember 2020 — Darmstadt — D 17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

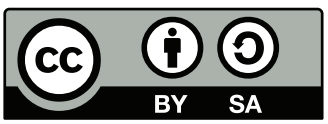
Robot Learning for Muscular Systems
Lernen auf Muskelbasierten Robotern

Genehmigte Dissertation von M.Sc. Dieter Büchler aus Duschanbe

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Tamim Asfour

Tag der Einreichung: 15.10.2019
Tag der Prüfung: 17.12.2019

Darmstadt — D 17



For my loving family.

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Bei der vorliegenden Dissertation stimmen schriftliche und elektronische Version überein.

Darmstadt, den 15. November 2019

_____

(Dieter Büchler)

Abstract

Today's robots are capable of performing many tasks that tremendously improve human lives. For instance, in industrial applications, robots move heavy parts very quickly and precisely along a predefined path. Robots are also widely used in agriculture or domestic applications like vacuum cleaning and lawn mowing. However, in more general settings, the gap between human abilities and what current robots deliver is still not bridged, such as in *dynamic tasks*. Like table tennis with anthropomorphic robot arms, such tasks require the execution of fast motions that potentially harm the system. Optimizing for such fast motions and being able to execute them without impairing the robot still pose difficult challenges that, so far, have not been met. Humans perform dynamic tasks relatively easy at high levels of performance. Can we enable comparable perfection on kinematically anthropomorphic robots?

This thesis investigates whether learning approaches on more human-like actuated robots bring the community a step closer towards this ambitious goal. Learning has the potential to alleviate control difficulties arising from fast motions and more complex robots. On the other hand, an essential part of learning is exploration, which forms a natural trade-off with robot safety, especially at dynamic tasks. This thesis's general theme is to show that more human-like actuation enables exploring and failing directly on the real system while attempting fast and risky motions.

In the first part of this thesis, we develop a robotic arm with four degrees of freedom and eight pneumatic artificial muscles (PAM). Such a system is capable of replicating desired behaviors as seen in human arm motions: 1) high power-to-weight ratios, 2) inherent robustness due to passive compliance and 3) high-speed catapult-like motions as possible with fast energy release. Rather than recreating human anatomy, this system is designed to simplify control than previously designed pneumatic muscle robots. One of the main insights is that a simple PID controller is sufficient to control this system for slow motions accurately. When exploring fast movements directly on the real system, the antagonistic actuation avoids damages to the system. In this manner, the PID controller's parameters and additional feedforward terms can be tuned automatically using Bayesian optimization without further safety considerations.

Having such a system and following our goal to show the benefits of the combination of learning and muscular systems, the next part's content is to learn a dynamics model and use it for control. In particular, the goal here is to learn a model purely from data as analytical models of PAM-based robots are not sufficiently good. Nonlinearities, hysteresis effects, massive actuator delay, and unobservable dependencies like temperature make such actuators' modeling especially hard. We learn probabilistic forward dynamics models using Gaussian processes and, subsequently, employ them for control to address this issue. However, Gaussian processes dynamics models cannot be set-up for our musculoskeletal robot as for traditional motor-driven robots because of unclear state composition, etc. In this part of the thesis, we empirically study and discuss how to tune these approaches

to complex musculoskeletal robots. For the control part, introduce Variance Regularized Control (VRC) that tracks a desired trajectory using the learned probabilistic model. VRC incorporates the GP's variance prediction as a regularization term to optimize for actions that minimize the tracking error while staying in the training data's vicinity.
In the third part of this thesis, we utilized the PAM-based robot to return and smash table tennis balls that have been shot by a ball launcher. Rather than optimizing the desired trajectory and subsequently track it to hit the ball, we employ model-free Reinforcement Learning to learn this task from scratch. By using RL with our system, we can specify the table tennis task directly in the reward function. The RL agent also applies the actions directly on the low-level controls (equivalent to the air pressure space) while robot safety is assured due to the antagonistic actuation. In this manner, we allow the RL agent to be applied to the real system in the same way as in simulation. Additionally, we make use of the robustness of PAM-driven robots by letting the training run for 1.5 million time steps ($14\,\mathrm{h}$). We introduce a semi sim and real training procedure in order to avoid training with real balls. With this solution, we return 75% of all incoming balls to the opponent's side of the table without using real balls during training. We also learn to smash the ball with an average ball speed of $12\,\mathrm{m\,s^{-1}}$ ($5\,\mathrm{m\,s^{-1}}$ for the return task) after the hit while sacrificing accuracy (return rate of 29%).
In summary, we show that learning approaches to control of muscular systems can lead to increased performance in dynamic tasks. In this thesis, we went through many aspects of robotics: We started by building a PAM-based robot and showed its robustness and inherent safety by tuning control parameters automatically with BO. Also, we modeled the dynamics and used this model for control. In the last chapter, we on top used our system for a precision-demanding task that has not been achieved before. Altogether, this thesis makes a step towards showing that good performance in dynamic tasks can be achieved *because* and not *despite* PAM-driven robots.

Zusammenfassung

Heutige Roboter sind in der Lage, viele Aufgaben zu übernehmen, die das menschliche Leben enorm verbessern. In industriellen Anwendungen beispielsweise transportieren Roboter schwere Teile sehr schnell und präzise auf einer vordefinierten Trajektorie. Roboter werden immer häufiger in der Landwirtschaft oder im Haushalt eingesetzt, wie zum Beispiel beim Staubsaugen oder Rasenmähen. Für generelle Aufgaben klafft jedoch noch eine Lücke zwischen menschlichen Fähigkeiten und dem, was heutige Roboter im Stande sind zu leisten. Ein gutes Beispiel dafür sind *dynamische Aufgaben*, wie Tischtennis mit anthropomorphen Roboterarmen. Solche Aufgaben erfordern die Ausführung von schnellen Bewegungen, die das System beschädigen können. Die Berechnung schneller Bewegungen und deren Ausführung ohne den Roboter zu gefährden, stellen eine große Herausforderung dar, die bisher nicht erfüllt wurde. Im Vergleich dazu sind dynamische Aufgaben relative leicht für Menschen zu erlernen und durchzuführen. Können wir ein vergleichbares Level mit anthropomorphen Robotern erreichen?

In dieser Arbeit untersuchen wir, ob Lernansätze angewendet auf menschenähnlich angetriebenen Roboter, uns diesem ehrgeizigen Ziel einen Schritt näher bringen können. Lernen hat das Potenzial, schwierige Regelungsprobleme zu lösen, die durch schnelle Bewegungen und komplexe Roboter entstehen. Ein essentieller Teil jedes Lernalgorithmuses besteht darin zu auszuprobieren und daraus zu lernen. Exploration kann auf realen Robotern gefährlich sein und bedarf deshalb einer Abwägung gegen die Sicherheit des Roboters, insbesondere bei dynamischen Aufgaben. Ein generelles Ziel dieser Arbeit ist es zu zeigen, dass menschenähnlichere Antriebe für Roboter es ermöglichen, direkt auf realen Systemen schnelle Bewegungen auszuprobieren und zu scheitern, um daraus lernen zu können.

Im ersten Teil dieser Arbeit, entwickeln wir einen Roboterarm mit vier Freiheitsgraden und acht pneumatischen künstlichen Muskeln (PAM). Dieses System ermöglicht es erwünschte Eigenschaften menschlicher Armbewegungen in dynamischen Aufgaben zu reproduzieren: 1) hohes Kraft-zu-Gewicht Verhältnis, 2) inhärente Robustheit durch passive Steifigkeit und 3) schnelle katapult-artige Bewegungen, wie sie durch schnelle Energiefreisetzung möglich sind. Im Kontrast zu bisher gebauten Robotern mit pneumatischem Muskelantrieb, wurde dieses System entwickelt, um die Regelung und Steuerung zu vereinfachen anstatt die menschliche Anatomie nachzubilden. Eine der wichtigsten Erkenntnisse, die wir dabei gewonnen haben, ist, dass ein einfacher PID-Regler ausreicht, um dieses System für langsame Bewegungen präzise zu steuern. Bei der Ausführung schneller Bewegungen direkt auf dem realen System hilft der antagonistische Muskelantrieb Schäden am System zu vermeiden. Auf diese Weise können die Parameter des PID-Reglers und zusätzliche Vorwärtsterme durch Bayes'sche Optimierung ohne weitere Sicherheitseinschränkungen automatisch optimiert werden.

Auf dem Weg, die Vorteile der Kombination von Lernansätzen und muskelbasierten Systemen aufzuzeigen, besteht der Inhalt des nächsten Kapitels darin ein Dynamikmodell zu lernen und dieses zur Regelung zu verwenden. Insbesondere geht es hier darum, ein Modell ausschließlich aus Daten zu lernen, da analytische Modelle von PAM-basierten Robotern

nicht gut genug sind. Gründe, warum die Ableitung von Modellen aus der Physik schwierig ist, sind Nichtlinearitäten, Hystereseeffekte, massive Stellgliedverzögerungen und schwer beobachtbare Abhängigkeiten wie z.B. von der Temperatur. Um dieses Problem anzugehen, lernen wir probabilistische Vorwärtsdynamikmodelle mit Hilfe von Gaußschen Prozessen und setzen sie anschließend zur Steuerung ein. Allerdings können Gaußsche Dynamikmodelle für muskel-basierte Roboter nicht wie für herkömmliche motorgetriebene Roboter eingesetzt werden, da beispielsweise die Zustandszusammensetzung unklar ist. In diesem Teil der Arbeit untersuchen wir empirisch und diskutieren im Detail, wie man diese Ansätze auf komplexe muskelbetriebene Roboter abstimmen kann. Zusätzlich stellen wir die Methode Variance Regularized Control (VRC) vor, die eine gewünschte Trajektorie mithilfe des erlernten probabilistischen Modells nachführt. VRC nutzt die Varianzvorhersage als Regularisierung, um den Nachführfehler zu minimieren während gleichzeitig das System in der Nähe der Trainingsdaten gehalten wird.
Im dritten Teil dieser Arbeit lernen wir Tischtennisbälle, die von einer Ballmaschine geworfen werden, auf den Tisch zurückzuspielen und zu schmettern. Anstatt eine Trajektorie des Schlägers zu optimieren, die den fliegenden Ball zurückspielen würde, und anschließend mit dem Roboter nachzuführen, setzen wir modellfreies Reinforcement Learning (RL) ein und lernen diese Aufgabe ohne Vorwissen einzusetzen. Der Vorteil dieses Ansatzes ist es, dass wir das wesentliche Ziel im Tischtennis direkt in der Belohnungsfunktion formulieren können, anstatt zu versuchen die berechnete Trajektorie als Ganzes nachzuverfolgen. Darüber hinaus wendet der RL-Agent seine Aktionen direkt auf die Low-Level-Steuerung (entspricht dem Luftdruck) an, während die Unversehrtheit des Roboters durch den antagonistische Muskelantrieb gewährleistet wird. Auf diese Weise kann der RL-Agent auf dieselbe Art und Weise in Simulation und dem realen System agieren. Darüber hinaus nutzen wir die Robustheit von PAM-gesteuerten Robotern, um das Training für 1,5 Millionen Zeitschritte auszuführen (entspricht etwa $14\,\mathrm{h}$). Um ein unpraktisches Training mit realen Bällen zu vermeiden, führen wir eine teil-simulierte und teil-reale Trainingsprozedur ein. Mit dieser Lösung retournieren wir 75% aller Bälle auf die Seite des Gegners, ohne vorher echte Bälle während des Trainings zu verwenden. Dabei lernen wir den Ball mit einer durchschnittlichen Ballgeschwindigkeit von $12\,\mathrm{m\,s^{-1}}$ ($5\,\mathrm{m\,s^{-1}}$ für das Zurückspielen) zu schmettern, was mit einer geringeren Genauigkeit einhergeht (29% der Bälle werden auf die andere Tischseite zurückgespielt).
Zusammenfassend zeigen wir in dieser Dissertation, dass Lernansätze zur Steuerung von Muskelsystemen hilfreich bei dynamischen Aufgaben sind. Dabei arbeiteten wir an vielen Aspekten der Robotik: Wir begannen mit der Entwicklung eines PAM-basierten Roboters und zeigten seine Robustheit, indem wir die Regelparameter automatisch mit Bayes'scher Optimierung ohne Sicherheitsbeschränkungen optimierten. Des Weiteren haben wir die Dynamik des Muskelroboters probabilistisch modelliert und dieses Modell unter Berücksichtigung der Varinzvorhersage zur Steuerung verwendet. Im letzten Kapitel, nutzten wir unser System, um eine dynamische Aufgabe zu lösen, die so bisher noch nicht erreicht wurde. Alles in allem, zeigt diese Arbeit, dass gute Lösungen für dynamischen Aufgaben erzielt werden können nicht *obwohl*, sondern *weil* muskelbasierte Systeme eingesetzt wurden.

Acknowledgments

This thesis has only been possible due to the support of several people. First and foremost, I would like to thank my PhD supervisor Prof. Jan Peters who has given me valuable advice not only in research, time management and writing, but he also shared his experience from various situations to help me do the right decisions. It is an honor and pleasure being mentored by Jan.
Likewise, I have learned a lot from Roberto Calandra. It has been a pleasure to work on several papers with Roberto and exchange research ideas with him. Working with Roberto always reminded me why I love doing research.
I want to express a big thanks to all the colleagues and to the staff of the Empirical Inference department at the Max Planck Institute for Intelligent Systems with whom I have worked over the years. Prof. Bernhard Schölkopf has created a lively environment that enables outstanding research to happen. Special thanks go to the Robot Learning Lab's former and current members: Yanlong Huang, Okan Koç, Sebastian Gomez-Gonzalez, and Simon Guist. You all have contributed directly or indirectly to the results of this thesis. Also, many thanks to Simon Guist and Nico Gürtler for proofreading parts of this thesis.
I would like to thank Prof. Tamim Asfour for reviewing this thesis and agreeing to be an external committee member. Your input is greatly appreciated. I would also like to thank the other members of my thesis committee, Professors Reiner Hähnle, Oskar von Stryk, Kristian Kersting, André Seyfarth and Jan Peters for investing the time to analyze my work. Your feedback is very welcome!
Finally, I am lucky to have friends and family who always support and motivate me to improve myself. My mother Erna and sister Lisa have always provided me with unconditional love. On a special note, I'm grateful for my wife, Yulia. No words seem to be enough to describe how thankful I am to have you in my life. You always stand next to me through the good and challenging times.

Contents

1 Introduction

Throwing and catching balls, playing football or table tennis seem natural and easy to learn for humans. Making real robots perform such tasks as agile as humans appears to be straightforward to society, as often pictured in movies. What triggers these expectations? The rationale could be that the pure existence of human-sized robots and the fact that robots are indeed exceeding human abilities at some tasks, like in manufacturing, let people extrapolate to any other task. Another possible reason is the recent success of Artificial Intelligence (AI) and Machine Learning (ML) algorithms in overcoming humans in complex games like Go [1, 2] as well as in the field of computer vision and image processing.
A more realistic view of current robotic systems is that robots surpass humans in force, repeatability, and precision while working almost 24/7 and hence perform well in repetitive and fully observable tasks. However, the performance suffers in more general settings where manually predefining solutions is not possible. For example, a robot can be easily programmed to return a table tennis ball to the other side of the table if the incoming ball trajectory is always similar. Being able to return balls where the trajectory substantially changes in speed as well as position requires more flexible and possibly fast movements of the robot. The combination of computing such movements *and* being able to execute them precisely is still a significant challenge in robotics. Especially, executing highly-accelerated motions in a way that helps to achieve high performance in a task, such as smashing a table tennis ball, is extraordinarily difficult.
We term the set of tasks with such demands *dynamic tasks*. Such problems are defined by

- **uncertainty** about the environment, e.g., the ball can be occluded, the sensors are corrupted by noise or other unobserved dependencies,
- **dynamically changing** environments, e.g., the opponent plays the ball either fast or slow, adds spin, or alters the bouncing position every time
- requiring the robot to generate **fast movements**, e.g., changing rapidly from forehand to backhand.

Consequently, algorithms are forced to run in real-time, learn from noisy and ambiguous sensor data, and robots are required to perform fast movements while avoiding damages. Dynamic tasks can merely quickly be learned by humans but pose ample challenges to anthropomorphic robots.
Using non-anthropomorphic robots can alleviate some of the robotics issues at dynamic tasks. For instance, the Japanese company Omron achieved impressive results in adversarial ping-pong against humans [3]. They employed a delta robot and located it over the table. Consequently, the robot's inertia is small, and the robot's reach is relatively similar to the human. In this manner, Omron could generate fast and precise motions as control of such robots is easier than anthropomorphic robot arms. In our work, we decided to

utilize anthropomorphic robot arms as we choose to address all implicit problems of such systems alongside the challenges of dynamic tasks.
We believe that high performance in anthropomorphic dynamic tasks cannot be solely achieved algorithmically but requires eventually robots that share some of the human body's properties. For instance, highly accelerated flick movements can be observed in the human wrist motions during a table tennis serve to obscure the ball's spin. Although possible, such movements are hard to replicate on traditional rigid and motor-driven bodies unless the robot is built in a robust and hence heavy manner. Having hardware that is better suited for such motions is, thus, desirable. Pneumatic artificial muscles (PAM) offer many beneficial properties. PAMs are inherently light and powerful, which allows generating fast motions with low inertia. Besides, PAMs are passively compliant actuators, making the joints backdrivable. Consequently, at impact with external objects, much of the stress is absorbed by the muscles. Moreover, variable stiffness due to antagonistic actuation of PAMs offers flexible solutions in various tasks. On the downside, using PAM-driven systems at dynamic tasks make the control substantially tricky. PAM-driven systems are nonlinear actuators, suffer from hysteresis, and their dynamics change with unobserved influences such as temperature. Approaches to learning control are a promising direction to help in such situations.
In this thesis, we built a PAM-driven robot arm that is suited for dynamic tasks and investigate learning approaches to use PAM-driven robots' extended capabilities better. In particular, we tackle the problems of modeling dynamics of this system, perform trajectory tracking tasks, and learn policies directly for table tennis. On this path, we investigate ML methods like Gaussian processes, Bayesian optimization, and Reinforcement Learning and incorporate methods from control theory.

1.1 Contributions

In this section, we outline the main contribution of this thesis in detail while addressing the key challenges.

1.1.1 Muscular Robot Design

Although robots actuated by PAMs inhere many beneficial properties for robotics, the control difficulties often render the application to precision-demanding tasks infeasible. As a result, systems driven by PAMs are not widely used, and precise control is only achieved on systems with up to one degree of freedom (DoF) and two PAMs. Nevertheless, robots have been built with complex kinematic structures and many DoFs. We aimed at creating a robot actuated by PAMs that is complex enough to perform interesting tasks while being as simple as possible to ease control. Thus we built a four DoF robot with the minimal amount of eight PAMs and used a lightweight arm with 700g moving masses [4, 5]. Also, we moved the PAMs to the base of the robot and avoided bending of cables and any other additional source of friction. As a proof of concept, a simple PID controller was enough to control our robot well compared to other PAM-based robotic arms.

1.1.2 Safe Exploration due to Antagonistic Actuation

The application of ML and especially Reinforcement Learning (RL) approaches on real systems always requires some way to achieve safe exploration [6]. Robot safety can be achieved in many ways. First, the robot itself can be underpowered and hence not able to reach high velocities and accelerations. Consequently, high performance in dynamic tasks like ball games is prohibited. Second, the torque on the motors can be limited by a constant threshold. The threshold is usually chosen conservatively; hence, areas in the state action space of good performance for a task are unnecessarily blocked. Third, on the algorithmic part, robust control that directly takes plant uncertainties into account can be incorporated, or exploration is guided to take actions that maximize performance while minimizing the risk of damages [7, 8]. All of these approaches rely on assumptions on the system, disturbances, or safety measures. When learning dynamic hitting movements, these assumptions can easily be violated. An antagonistic PAM pair, on the other hand, can assure safety by adjusting the allowed air pressure range of each muscle. In case the joint limits are almost reached, the antagonist pneumatic muscle to the direction of movement of the link will exert strong forces and stop the link before some predefined joint angle are reached. Thus, damages to the robot itself or any obstacles can be averted. As a result, we can apply classic Bayesian optimization techniques directly on the controller parameters without damaging the system and making fully use of the method's capabilities [5]. With our solution, an algorithm can be directly tested in muscle space, and any program that works in simulation can directly be transferred to the real system.

1.1.3 Gaussian Process Application to PAM Systems

Gaussian process (GP) [9] forward models are part of many modern learning control approaches like PILCO [10] and can also be used in model-based RL. A GP is a non-parametric regression method that expresses a distribution over functions. GPs define a covariance function that effectively imposes an assumption on the smoothness of the underlying function. The smoothness does not have to be set by hand but is optimized over the covariance function and the data. Thus GPs are a suitable and widely used regression technique to mimic nonlinear dynamics. Unfortunately, our system suffers, in addition to the issues reported above, from severe actuator delay and strong heteroscedastic state-dependent noise. A traditional GP, however, assumes no input noise and homoscedastic output noise. Unclear state representation and selection of training data points influence the prediction performance tremendously. Our contribution is to analyze how these design choices alter the prediction performance on data sets that are corrupted by high noise levels, heteroscedastic noise, and a fast data set that excites higher order dynamics [11]. With this analysis, GP dynamics models can now be applied to model PAM driven robots - which has not been done before - and leverage their probabilistic framework for control.

1.1.4 Control with Variance Regularization

A major difficulty for using GP forward models for control lies in the fact that non-parametric methods are only useful near their training data. Muscular systems inhere

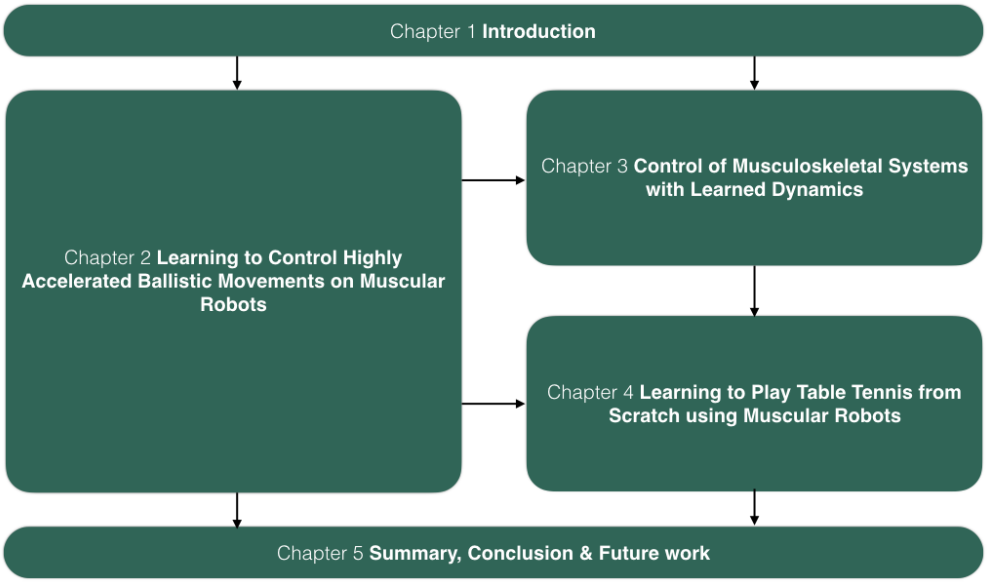


Figure 1.1: Structure of the thesis. Arrows indicate possible order of reading. The Introduction pictures what kind of problems are solved in this thesis. Chapter 2 describes the PAM-based arm that all experiments are performed on. From there, either chapter 3 or 4 can be easily read as they both enable the application of Machine Learning methods on muscular systems. Chapter 5 closes this thesis by wrapping up, concluding, and mentioning possible future work.

some properties that impede staying local in the state-action space. First, have a high state dimensionality as PAM-actuated robots always require at least two muscles per DoF. In addition to the robot arm dynamics state $\mathbf{s} = [\mathbf{q}, \dot{\mathbf{q}}]$, muscle lengths and length velocities $\mathbf{l}, \dot{\mathbf{l}}$ as well as air pressures $\mathbf{p}$ have to be incorporated. Hence, more data is required to fill the state-action space compared to traditionally actuated robots. In order to alleviate this problem, we make use of the probabilistic GP forward model by regularizing with the variance of the GP [11]. In this manner, we make use of the fact that an infinite set of air pressure combinations lead to the same joint angle by choosing the pressure combination that is closest to the training data.

1.1.5 Playing Table Tennis with Muscular Robots

Table tennis is a type of sport that humans learn relatively fast while replicating similar robots' performance is hard. To achieve a high level of dexterity in table tennis, the robot needs to exert high accelerations, which can be observed in human arm movements. Imagine that the ball is supposed to be hit at a far away location from the current racket position. In this case, the robot needs to accelerate the racket rapidly to gain momentum and hit the flying ball in time. Robots actuated pneumatic muscles are capable of generating a satisfying amount of acceleration for this task. The question is whether such robots, which are inherently hard to control, can be controlled accurately enough to return balls precisely. In [5], we use the robustness of PAM-driven systems to enable long-term training with model-free RL. In this manner, we learn to return balls shot by a ball launcher

reliably. Additionally, we learn to smash the ball while sacrificing precision compared to the return experiment. As the hardware inherently handles safety, we do not specify any further safety measures. On the contrary, we favor fast motions by maximizing the ball speed in the reward function.

1.2 Thesis Outline

This section presents the outline of the thesis and clarifies how the individual contributions fit together. The individual chapters of this thesis can be mainly read independently. Still, reading them in order gives a more in-depth understanding. The PAM-based robot's construction and experiments showing that exploration in dynamic motions is safe are described in Chapter 2. Chapter 3 and 4 mostly focus on the algorithmic side. All of the approaches developed in Chapters 3 and 4 are applied to the system from Chapter 2. In Chapter 5, we summarize this thesis's main contributions and discuss open problems and remaining challenges.

Chapter 2 presents our four DoF lightweight robotic arm that is actuated by eight PAMs. It introduces the core construction considerations compared to previously built arms actuated by PAMs. As a result, a simple PID controller achieves similar control performance as previously shown in other publications. Additional experiments illustrate that this system is very well suited to exploring dynamical motions without damaging the system.

Chapter 3 presents how to set up GP forward dynamics models for PAM-driven systems. Many issues arise when using muscular systems compared to traditionally actuated robots for model learning. These issues are identified with experiments, and solutions are proposed. Subsequently, the GP's variance is incorporated into control to stay in the vicinity of the training data.

Chapter 4 describes RL's application of learning to strike a table tennis ball to the opponent's side. We use a hybrid sim and real training procedure to enable long-term training without utilizing real balls. After training, the agent learned to return and smash real balls without touching a real ball during training.

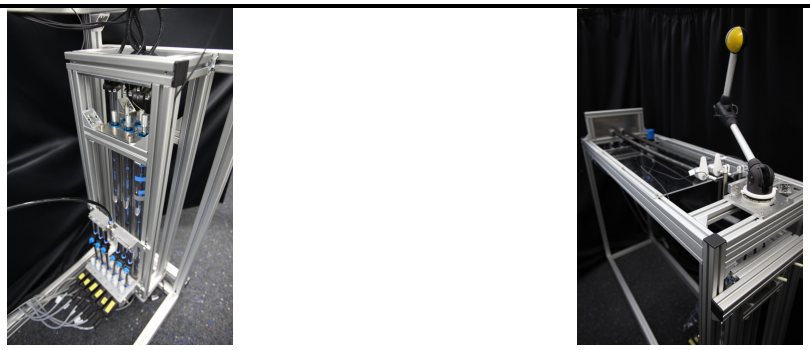
Chapter 5 summarizes our approach and presents the main conclusions of this thesis. Further, we discuss open challenges and the potential extensions of our approach.

2 Learning to Control Highly Accelerated Ballistic Movements on Muscular Robots

High-speed and high-acceleration movements are inherently hard to control. Applying learning to the control of such motions on anthropomorphic robot arms can improve the accuracy of the control but might damage the system. The inherent exploration of learning approaches can lead to instabilities and the robot reaching joint limits at high speeds. Having hardware that enables safe exploration of high-speed and high-acceleration movements is therefore desirable. To address this issue, we propose to use robots actuated by Pneumatic Artificial Muscles (PAMs). In this chapter, we present a four degrees of freedom (DoFs) robot arm that reaches high joint angle accelerations of up to $28\,000\,^\circ\mathrm{s}^{-2}$ while avoiding dangerous joint limits thanks to the antagonistic actuation and limits on the air pressure ranges. With this robot arm, we are able to tune control parameters using Bayesian optimization directly on the hardware without additional safety considerations. The achieved tracking performance on a fast trajectory exceeds previous results on comparable PAM-driven robots. We also show that our system can be controlled well on slow trajectories with PID controllers due to careful construction considerations such as minimal bending of cables, lightweight kinematics and minimal contact between PAMs and PAMs with the links. Finally, we propose a novel technique to control the the co-contraction of antagonistic muscle pairs. Experimental results illustrate that choosing the optimal co-contraction level is vital to reach better tracking performance. Through the use of PAM-driven robots and learning, we do a small step towards the future development of robots capable of more human-like motions.

2.1 Introduction

Controlling highly accelerated movements on anthropomorphic robot arms is an aspired ability. High accelerations lead to high velocities over a small distance which enables fast reaction times. Such motions can be observed in human arm trajectories, known as ballistic movements [12]. However, producing ballistic movements on robots is challenging because they 1) are inherently hard to control, 2) potentially run the joints into their limits and hence break the system and 3) require hardware that is capable of generating high accelerations. We use the term *high-acceleration tasks* to refer to the set of such problems. A promising way to approach problem 1) is to apply Machine Learning approaches - that inherently explore - to learn directly on the real hardware. In this manner, algorithms can automatically tune low-level controllers that, for instance, track a fast trajectory with lower control error than manually tuned controllers. Problem 2), however, currently rules this



(a) (b)

Figure 2.2: Hardware components of our robot designed to keep friction low. (a) 6 PAMs are located directly below the Igus arm in order to pull the cables in the same direction as they exit the arm so that deflection is minimized. The necessary bending of the cables is realized by Bowden cables. (b) 2 PAMs actuating the first DoF are located on top of the base frame. They are longer ($1\,\mathrm{m}$) than the other six PAMs ($0.6\,\mathrm{m}$) due to the bigger radius of the first rotational DoF.

path out and is even more problematic once we generate higher accelerations (Problem 3)).

Hence, enabling exploration in such fast domains by preventing potential damages from the hardware side, can help improve performance in high-acceleration tasks.

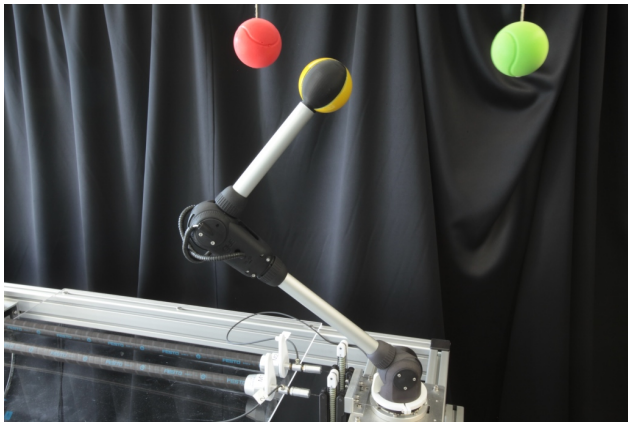


Figure 2.1: Igus Robolink lightweight arm with $700\,\mathrm{g}$ of moving masses. Eight powerful antagonistic PAMs move four DoFs where each joint contains two rotational DoFs. Rather than recreating human anatomy, our system is designed to ease control to facilitate the learning of fast trajectory tracking control. Experimental results show that our robot is precise at low speeds using a simple manually tuned PID controller while reaching high velocities of up to $12\,\mathrm{m\,s^{-1}}$ ($200\,\mathrm{m\,s^{-2}}$) in task space and $1500\,^\circ\mathrm{s^{-1}}$ ($28\,000\,^\circ\mathrm{s^{-2}}$) in joint space.

The human arm anatomy possesses many beneficial properties over current anthropomorphic motor-driven robotic arms for high-acceleration tasks. While motor-driven systems can generate high speeds, it is hard to produce high accelerations and keep the kinematics human arm sized at the same time. Instead, muscles drive the human arm. Skeletal muscles generate high forces and are located as close as possible to the torso to keep moving masses to a minimum. Concurrently, the human arm inhibits damage at collisions thanks to the built-in passive compliance which ensures deflection of the end-effector instead of breakage as a response to external forces.

Robotic arms actuated by antagonistic pneumatic artificial muscle (PAM) pairs own some of these desired abilities. In addition to high accelerations and compliance, PAMs exhibit similarities to skeletal muscles in static and dynamic behavior [29, 30, 31, 32]. However, PAMs do not fully resemble the skeletal muscle. PAMs pull only along their linear axes and break when curled. Muscle structures bending over bones like the deltoid muscles that connect the acromion with the humerus bone at the shoulder are hardly realizable. Furthermore, biological muscles can be classified as wet-ware whereas PAMs suffer from additional friction when touching each other or the skeleton during usage. Thus, bi-

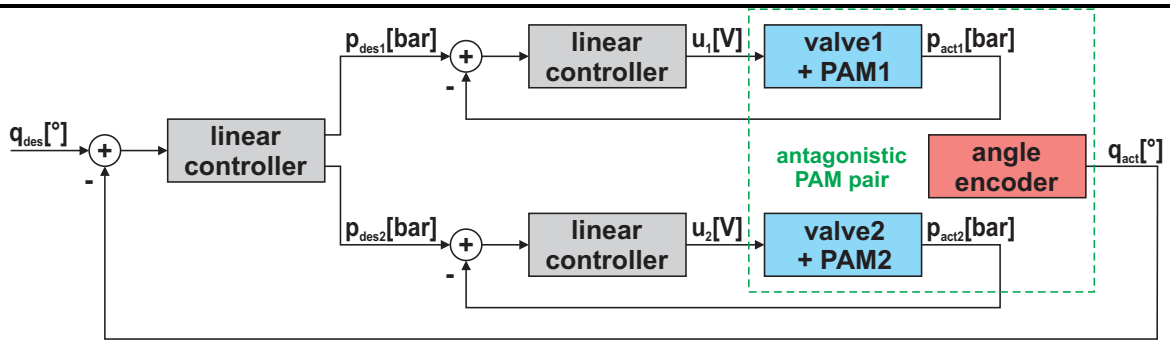


Figure 2.3: Schematic description of the position control loop for one PAM muscle pair. The absolute value of the output signal of the position control PID is assigned following the symmetrical co-contraction approach discussed in Section 2.3.2. The pressure within each PAM is governed by separate PIDs that set the input voltage to the proportional air flow valves. The sensor values are provided by Festo™ pressure sensors and angle encoders.

articular configurations (one PAM influences two DoFs) like the ones present in the human arm with seven DoFs are hard to realize. Although it seems that PAMs are well suited to be attached directly to the joints instead of using cables due to their high power-to-weight ratio, this results in bigger moving masses and, thus, more non-linearities.

Many systems have been designed with the aim of reproducing the human anatomy using PAMs (see Table 2.1). Although such recent publications show good tracking performance of one PAM in position [33, 34, 35], using PAM-based systems with more DoFs for fast trajectory tracking appears to be less satisfactory. The performance of PAM-actuated robots has thus been limited to slow movements compared to servo motor driven robots. In Table 2.1 we list, along with the existing PAM-actuated arms, the most complex (form and velocity) tracked trajectory in case it was mentioned. For our purpose it is crucial that anthropomorphism does not degrade the ease to control the resulting arm.

In this chapter, we present a robot (see Figure 2.1 and Figure 2.2) that fulfills our requirements while avoiding the problems of previous construction to achieve precise and *fast* movements. We illustrate the effectiveness of our hardware considerations by showing good results at tracking of slow trajectories with PIDs only. Additionally, we demonstrate high acceleration and velocity motions by applying step control signals to the PAMs. These motions surpass the peak velocity and acceleration of the Barrett WAM arm, that is used for robot table tennis [36, 37], by a factor of 4x and 10x respectively while being able to sustain the mechanical stress. Another contribution of this chapter is the tuning of control parameters using Bayesian optimization (BO) *without* any safety considerations. Although previous papers employed BO on real robots [38, 39, 40], the applications have been limited to rather slow and safe motions whereas we even allow for unstable controllers during training as long as the motion in bounded (see Section 2.3.1). Our path is parallel to the sensible approach of taking safety directly into account, such as by means of constraints [41], where we enable safety through antagonistic actuation for high-acceleration tasks. Using the parameters learned with BO, we track - to the best of our knowledge - the fastest trajectory that has been tracked with a four DoF PAM-driven arm. At last, we empirically show that choosing the appropriate co-contraction level is essential to achieve good control performance.

We encourage other researchers to use our platform as a testbed for learning control approaches. We used off-the-shelf and affordable parts like PAMs by Festo™, the robot arm by

Igus and build the base using Item profiles. All necessary documents to rebuild our system and videos of its performance can be found at `http://musclerob.robotlearning.ai`.

2.2 System Design to Generate and Sustain Highly Accelerated Movements

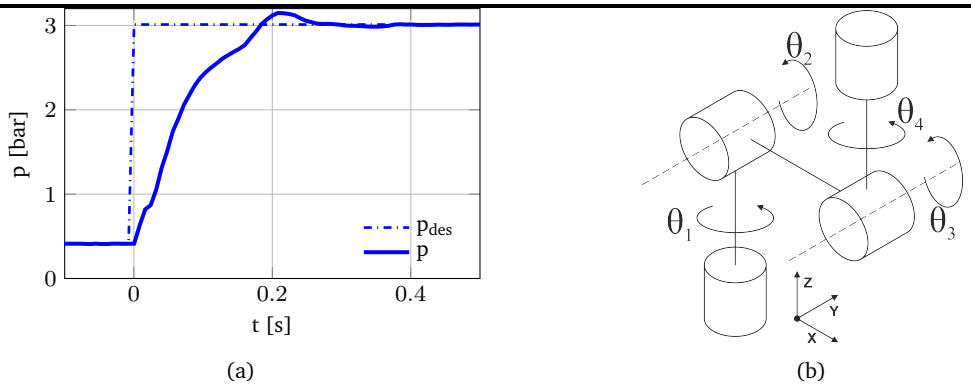
Using systems actuated by PAMs can improve performance at high-accelerations tasks on real robots by applying Machine Learning to tune low-level controller. On the other hand, such systems add additional control challenges. We identify the following key opportunities to ease control of such systems: 1) avoiding friction between muscles, 2) avoiding contact between muscles and skeleton, 3) installing PAMs in the torso to decrease moving mass, 4) minimal deflection of cables, 5) light-weight segments, 6) mostly independent DoFs. These points guided the construction of our four DoF PAM-driven arm.

2.2.1 Igus™ Robolink Lightweight Kinematics

To achieve high accelerations, it is generally desirable to have low moving masses. At the same time, minimizing the weight also minimizes the non-linearities within a system (especially the weight at the end-effector). Hence, we incorporate a light-weight tendon-driven arm by Igus™ [42] that has four DoFs and is actuated by eight PAMs (two PAMs per DoF, Figure 2.3). The arm has two rotational DoFs in each of the two joints and weighs less than $700\,\mathrm{g}$ in total. The first joint, which is fixed to the base, contributes little to the moving mass. As a result, the PAM dynamics are dominant over the arm dynamics. Besides, it is driven by Dyneema tendons (2mm diameter, tensile strength of $4000\,\mathrm{N}$) that allow fixing the PAMs in the base. Necessary deflections within the Igus™arm are realized through Bowden cables. They guide the cables within the arm almost without influencing each other and keep the length unchanged during movement. As a result, cross-talking between DoFs due to cables is minimized. Still, little cross-talking persists as the PAMs share the same air pressure supply as well as due to the non-zero moving mass.
Cable-driven systems usually suffer from additional friction. For this reason, the tendons are only minimally bent by our construction. All PAMs pull their respective tendons in the same direction as they exit the Igus™ arm. Two PAMs actuate the first rotational DoF in the base joint in the horizontal orientation, whereas the other 6 PAMs pull in the vertical direction, as can be seen in Figure 2.2a and b, respectively. Angular encoders measure the joint angles with a resolution of approximately $0.07°$. The kinematic structure is depicted in Figure 2.4b.

2.2.2 Software Framework

The complete system comprises eight pressure sensors and proportional valves as well as four incremental angular encoders to govern and sense the movement. Each DoF is actuated by two antagonistically aligned PAMs. The contraction ratio as well as the pulling force is influenced by the air pressure within each PAM. Thus, a low level controller regulates the pressure within each PAM using Festo proportional valves as can be seen in Figure 2.3. As a result, the control algorithm that regulates the movement works on top



(a) (b)

Figure 2.4: (a) Pressure step response from minimum to maximum value of 3 bar. The desired pressure value can be reached within approximately 250 ms. (b) Kinematic structure of the Igus™ Robolink arm. Two rotational DoFs are located at the base (θ_1 and θ_2) and two additional in the second joint (θ_3 and θ_4).

of these and sends desired pressures $\mathbf{p}^{\text{des}}$ to control the joint angles $\mathbf{q}$. A National Instruments PCIe 7842R FPGA card has been used to take over low level tasks such as extraction of the angular values from the A und B digital signals given by the encoder or regulating the pressure within each PAM. The FPGA was programmed in Labview. To assure fast implementation, we used the FPGA C/C++ API interface to generate a bitfile along with header files which can be incorporated in any C++ project. Thus, the control algorithm can be implemented in C++ on top of the basic functionalities supplied by the FPGA. The sensor values are read at $100\,\text{kHz}$ and new desired pressure values are adjusted at $100\,\text{Hz}$. Figure 2.4a shows the pressure response to a step in desired value from minimum ($0\,\text{bar}$) to maximum air pressure ($3\,\text{bar}$). The resulting pressure regulation reaches the desired value within a maximum of $0.25\,\text{s}$.

2.2.3 Reasons to Use Pneumatic Artificial Muscles

Apart from lightweight kinematics, generating high accelerations requires high forces. Hence, we use PAMs by Festo™ to actuate our robotic arm. PAMs consists of an inner rubber tube surrounded by a braided weave composed of repeated and identical rhombuses. An increase in air pressure leads to a gain in the diameter of the inner balloon. The double-helix-braided sheave transforms the axial elongation into a longitudinal contraction. This contraction process can be fully characterized according to the inner tube's radius and the braid angle. The inner pressure plays the same role as the neuronal activation level of a biological muscle. The dynamics of both PAMs and biological muscles share some characteristics that are captured to some extent by the Hill muscle model [29, 30, 31]

$$(F + a)(V + b) = b(F_0 + a)\,, \tag{2.1}$$

where F and V are the tension and contraction velocity of the muscle, a and b muscle-dependent empirical constants and F_0 the maximum isometric force generated in the muscle.
In our robot, two $1\,\text{m}$ and six $0.6\,\text{m}$ PAMs, each with a diameter of $20\,\text{mm}$, actuate four DoFs ($M = 4$, two PAMs per DoF). Each PAM can generate maximum forces of up to $1200\,\text{N}$ at $6\,\text{bar}$. We limit the pressure to a maximum of $3\,\text{bar}$ because the generated

accelerations are sufficient and to prolong the lifetime of the system. Figure 2.4a shows that the maximum desired pressure can be reached within appr. $250\,\mathrm{ms}$. 2.2.2 describes the software framework in detail.
Using PAMs to actuate robots comes with beneficial properties. The high forces of PAMs better overcome the resisting force of static friction. Also, fast and catapult-like movements can be generated by pressurizing both PAMs and discharging one of them. A similar kind of energy storage and release can also be found in human and primate arms in ballistic movements. At such high velocities, antagonistic muscle actuation can also be used to physically sustain high stress as well as ensure to stay within predefined joint ranges. Robot safety can be achieved through pretensioning each PAM with an individual minimum pressures $\mathbf{p}^{\min} \in \mathbb{R}^{2M}$ and, at the same time, set a maximum pressure to each PAM $\mathbf{p}^{\max} \in \mathbb{R}^{2M}$. In this manner, a fast motion can be stopped before exceeding the joint limits as the torque generated by the agonist PAM is upper-bounded by not exceeding $\mathbf{p}^{\max}$ and the stopping torque of the antagonist muscle counteracting the motion is high enough decelerate in time. Unlike motors, PAMs produce exponentially higher tensile forces when being stretched for a constant control signal (desired pressure for PAMs, desired torques for motors). In this manner, our system stays in safe joint ranges for any pressure trajectory *including step signals* without any need for sophisticated treatment of robot safety as required for motor-driven systems. In Section 2.4.2, we show that our system generates high accelerations within this pressure range without internal damages. In particular, for motor-driven systems, the torque must be adapted close to dangerous configurations, whereas thresholding the pressures is sufficient for manipulators actuated by PAMs as we found empirically.
Despite advantageous properties, PAMs are hard to control, which is the reason why they are not widely adopted. The main reason for the hard control challenges is the lack of sufficiently good models. Deriving models from physics is tough due to 1) the non-linear relationship between length, contraction velocity, and pressure, 2) time-varying behavior (as a result of dependencies on temperature and wearing) as well as 3) hysteresis effects [28, 43]. The non-linear effects are, among other effects, introduced by the compressibility of the air, the soft elastic–viscous material, and the geometric properties [44]. Additionally, the valve dynamics add non-linearities as the set variable is the pressure, which changes with the contraction ratio. Hysteresis within the PAMs is caused by the friction between the braided strands [26]. Additional hysteresis effects occur due to stiction in the joints. These issues render modeling of PAM-driven systems even for data-driven approaches challenging [11]. Thus, PAMs have been mainly applied due to their safety properties and high power-to-weight ratio for slow movements with the ability to carry heavy objects. Still, using such a system as a testbed for learning control approaches is a promising direction.

2.3 Using Bayesian Optimization to Tune Control of Muscular System

Our system is capable of generating highly accelerated motions and allows to explore in such fast regimes. The obvious next step is to automatically tune control parameters directly on the hardware. We use Bayesian optimization (BO) to learn to track a fast and hitting-like trajectory. This section introduces briefly explains the optimization method

and PID control with feedforward compensation that is used to control this overpowered system. Also, we adapt the symmetrical co-contraction approach so that co-contraction can be part of the tuning.

2.3.1 Overpowered System Control

Tuning feedback controller for PAM-driven system is hard due to actuator delay, unobserved dependencies, non-linearities and hysteresis effects [11, 43]. Still, well-tuned linear Proportional Integral Derivative (PID) controllers are often sufficient to track slow trajectories (see Section 2.4.1). The underlying control law

$$\mathbf{u}_t = \mathbf{u}_t^{\text{fb}} + \mathbf{u}_t^{\text{ff}}\,, \tag{2.2}$$

consists of a feedback $\mathbf{u}_t^{\text{fb}} \in \mathbb{R}^M$ and can be extended by a feedforward part $\mathbf{u}_t^{\text{ff}} \in \mathbb{R}^M$ where M represents the number of DoFs. The PID feedback controller

$$\mathbf{u}_t^{\text{fb}} = K^{\text{P}}\tilde{\mathbf{q}}_t + K^{\text{D}}\dot{\tilde{\mathbf{q}}}_t + K^{\text{I}}\tilde{\mathbf{q}}_t^s\,, \tag{2.3}$$

takes the position $\tilde{\mathbf{q}}_t = \mathbf{q}_t - \mathbf{q}_t^{\text{des}} \in \mathbb{R}^M$, velocity $\dot{\tilde{\mathbf{q}}}_t = \dot{\mathbf{q}}_t - \dot{\mathbf{q}}_t^{\text{des}} \in \mathbb{R}^M$ and integral errors $\tilde{\mathbf{q}}_t^s \in \mathbb{R}^M$ as input where $[\tilde{\mathbf{q}}_t^s]_i = \int_0^t \tilde{q}_i(x)dx$. The integral part compensates for steady-state errors. PIDs can be tuned by optimizing the elements of the feedback gain matrices $K^{x_{\text{fb}}} = \text{diag}(k_1^{x_{\text{fb}}}, \ldots, k_M^{x_{\text{fb}}})$ with $x_{\text{fb}} \in \{\text{P,I,D}\}$. The position feedback is always delayed by at least one cycle and hence subject to instabilities.
Feedforward compensation

$$\mathbf{u}_t^{\text{ff}} = K^{\text{pos}}\mathbf{q}_t^{\text{des}} + K^{\text{vel}}\dot{\mathbf{q}}_t^{\text{des}} + K^{\text{acc}}\ddot{\mathbf{q}}_t^{\text{des}}\,, \tag{2.4}$$

instantly generates a control signal in response to the current desired joint position $\mathbf{q}_t^{\text{des}}$, velocity $\dot{\mathbf{q}}_t^{\text{des}}$ and acceleration $\ddot{\mathbf{q}}_t^{\text{des}}$. In accordance to the feedback gain matrices, the feedforward gain matrices are also diagonal $K^{x_{\text{ff}}} = \text{diag}(k_1^{x_{\text{ff}}}, \ldots, k_M^{x_{\text{ff}}})$ with $x_{\text{ff}} \in \{\text{pos,vel,acc}\}$. Feedforward compensation has many beneficial properties. First, feedforward terms can help to reduce the malicious effects of hysteresis. Second, feedforward terms do not affect the stability of the feedback part [45], hence can be used purely to improve tracking performance. Third, tracking capabilities of pure feedback controllers are unavoidably degenerated for trajectories with high values of speed and accelerations. Feedforward terms help in such situations as the reaction to a desired fast motion happens instantly and is not delayed by at least one cycle as in feedback control.
The fact that the muscle dynamics are dominant over the rigid body dynamics has consequences for how we control the robot. PAMs are dominant because they generate high forces while the arm is lightweight. For this reason, we decided to perform independent joint control, thus, we chose the gain matrices $K^{x_{\text{c}}}$ where $x_{\text{c}} \in \{\text{P,I,D,pos,vel,acc}\}$ to be diagonal although they generally have non-zero off-diagonal elements. Another consequence of the dominant PAM dynamics is that the non-linearities due to the rigid body dynamics are less effective. Still, the PAM dynamics are non-linear and the linear dynamics assumption of PID controller with linear feedforward terms cannot be fulfilled.

Fortunately, PIDs work well even for approximately linear systems. For this reason, we assume the parameters to be valid in the vicinity of a specific trajectory rather than for arbitrary desired motions. In Section 2.4.3, we validate this claim by finding parameters that lead to good tracking performance on a fast trajectory.
Some sets of parameters for PID controllers lead to instabilities. Instabilities can cause damages by 1) running the links into their particular joint limits with high velocities and make the robot hit itself or by 2) creating high internal forces that break parts inside the robot such as connections to cables. For the latter case, we show in Section 2.4.2 that our system sustains step pressure signals that generate a highly accelerated and fast motion. The high internal forces created by such a motion did not damage any internal parts, most probably because the PAMs are backdrivable. The first case is more complicated: Our system does not break for periodic motions caused by instable controllers as long as the motion is bounded, a term we coin as *bounded instabilities*. In other words, the control signal does not excite the resonance frequency of the motion and add energy with every period. The bound for the periodic motion can be as wide as the allowed joint limits for each DoF. From our experience, it is sufficient to adjust an upper limit to the elements of the feedback gain matrices. If an unbounded instability occurs, the robot can be stopped by releasing the air from the PAMs and manually holding the robot. The low inertia due to the low moving masses as well as the backdrivable PAMs enables to proceed in this manner. This procedure is not possible on traditional motor-driven systems as the higher inertia can cause higher forces at impact and the motors are usually not backdrivable. Additionally, traditional systems would break due to high internal forces generated at such behaviors. In Section 2.4.3 we tune PID parameter with BO without further safety considerations on a fast trajectory while allowing bounded instabilities.

2.3.2 Adapted Symmetrical Co-contraction Approach

An antagonistic PAM pair is a multi-input-single-output-system (MISO) where both PAMs p_a and p_b influence the joint angle q. In contrast, a controller for a traditionally motor-driven robot outputs a scalar control signal u for each DoF. In our case, this scalar control signal has to map to both desired pressure for both PAMs p_a^{des} and p_b^{des} of an antagonistic pair to form a single input single output (SISO) system. One way is to assign the control signal in opposing directions to each PAM

$$p_x = p_0 \pm u\,, \tag{2.5}$$

as suggested in [24] where $x \in \{a, b\}$. Here, the assumption is that the joint angle q of each DoF increases with rising p_a and falling p_b and vice versa. The co-contraction parameter p_0 correlates with the stiffness of the corresponding DoF. However, the input range for the control signal $[u_{\text{min}}, u_{\text{max}}]$ for which at least one of the pressures p_a and p_b change depends on the value of p_0 as can be seen in Figure 2.5a. A fixed input range is essential in case p_0 should be optimized for control next to the elements of the

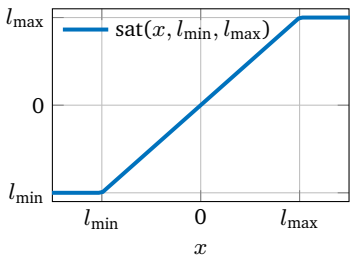


Figure 2.6: Saturation function

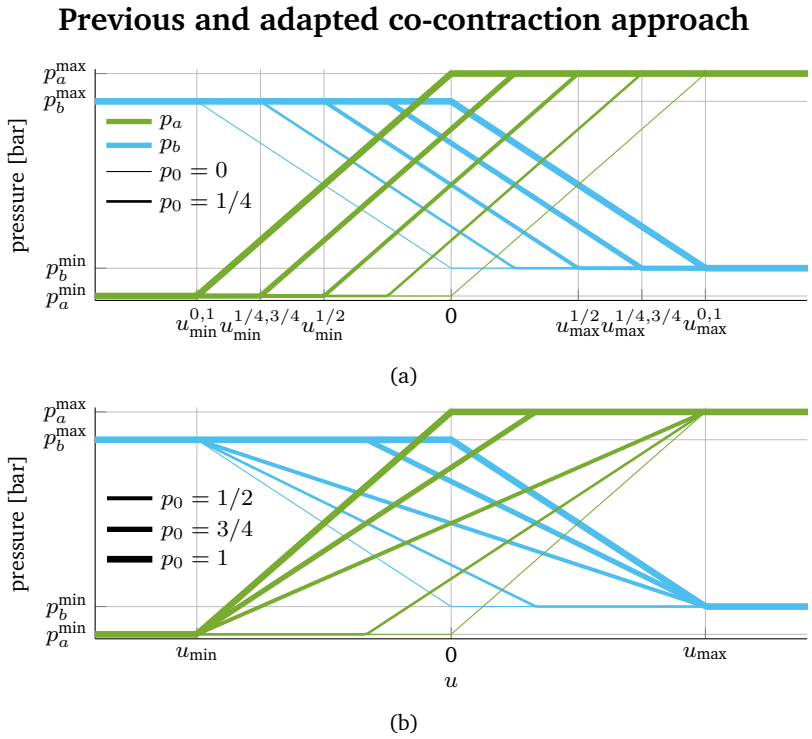


Figure 2.5: Approach to assign both pressures p_a and p_b of an antagonistic PAM pair from a scalar control signal u, hence, converting from a MISO into a SISO system. The thickness of the lines indicate different p_0 values defined in Equation 2.5 and Equation 2.6. The two colors represent p_a and p_b respectively. The sum of p_a and p_b increases with increasing p_0, thus, increasing the stiffness in the antagonistic PAM pair. (a) Symmetrical pressure approach from Equation 2.5 with additional saturation to keep p_a and p_b within the allowed ranges. The control range $[u_{\min}, u_{\max}]$ that effectively changes at least p_a or p_b changes for varying p_0 where the superscript indicates p_0 for the respective control range (e.g. $u_{\min}^{0,1}$ stands for lower range limit for $p_0 = 0$ and $p_0 = 1$). (b) Approach that corrects for changing effective control ranges for varying p_0 by adapting the slope within $[u_{\min}, u_{\max}]$ with c (see Equation 2.6).

gain matrices from Equation 2.3 and Equation 2.4. We extend this approach by first allowing the scalar control signal u to be only within $[-1, 1]$ and, secondly, linearly map the resulting number to an allowed pressure range

$$p_x = (p_x^{\text{max}} - p_x^{\text{min}})\big(p_0 \pm c\,\text{sat}(u, -1, 1)\big) + p_x^{\text{min}}\,, \tag{2.6}$$

where p_x^{max} and p_x^{min} with $x \in \{a, b\}$ are the individual maximal and minimal pressures. The saturation function $\text{sat}(\cdot, l_{\text{min}}, l_{\text{max}})$ is depicted in Figure 2.6 with l_{min} and l_{max} being the lower and upper threshold. The saturation function that keeps u withing the range $[-1, 1]$ in Equation 2.6 ensures that the computed desired pressures stay within the allowed ranges. Additionally, we added a correction parameter $c = 0.5 - \text{sat}(|p_0 - 0.5|, 0, 0.5)$ with the absolute value $|\cdot|$, to create different slopes depending on the value of p_0. Figure 2.5b depicts our corrected solution.

2.3.3 Bayesian Optimization

Bayesian optimization (BO) is a zero-order optimization technique [46, 47, 48] that aims at finding the global minimum

$$\mathbf{x}^* = \underset{\mathbf{x}}{\text{argmin}}\, f(\mathbf{x})\,, \tag{2.7}$$

of an unknown function $f : \mathbb{R}^D \to \mathbb{R}$ with inputs $\mathbf{x} \in \mathbb{R}^D$. BO operates in a black-box manner as the function is not required in analytical form but rather is modeled as a response surface $\tilde{f}$ from samples collected in a dataset $\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i))|i = 0, 1, \dots N - 1\}$ of the input parameters $\mathbf{x}_i \in \mathbb{R}^D$ and the resulting function evaluation $f(\mathbf{x}_i) \in \mathbb{R}$. Often probabilistic regression techniques are incorporated to handle noisy observations in a principled way, take model uncertainties into account and allow to integrate domain knowledge using priors. Among other methods, Gaussian Processes (GPs [9]) are widely used. A GP is a distribution over functions where the conditional posterior distribution is Gaussian

$$p(\tilde{f}|X, \mathbf{y}, \mathbf{x}_*) = \mathcal{N}(\mu, \sigma^2)\,, \tag{2.8}$$

with mean and variance

$$\mu(\mathbf{x}_*) = \mathbf{k}_*^T (K + \sigma_n^2 I)^{-1} \mathbf{y}\,, \tag{2.9}$$

$$\sigma^2(\mathbf{x}_*) = k_{**} - \mathbf{k}_*^T (K + \sigma_n^2 I)^{-1} \mathbf{k}_*\,, \tag{2.10}$$

where $X \in \mathbb{R}^{N \times D}$ is a design matrix with each row being the n-th training input $\mathbf{x}_n^T \in \mathbb{R}^{1 \times D}$, $\mathbf{y} \in \mathbb{R}^D$ are the target values, $[K]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, $[\mathbf{k}_*]_i = k(\mathbf{x}_i, \mathbf{x}_*)$, $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$ and I is the identity matrix. The function $k(\mathbf{x}_a, \mathbf{x}_b)$ is a kernel that represents the correlation between two data points $\mathbf{x}_a$ and $\mathbf{x}_b$. Here, we consider the Matérn 5 kernel with Automatic Relevance Determination (ARD)

$$k(\mathbf{x}_a, \mathbf{x}_b) = \sigma_f \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) \exp(-\sqrt{5}r) + \sigma_n \delta_{a,b}\,, \tag{2.11}$$

where $r^2 = \sum_{d=0}^{D-1}(x_{ad} - x_{bd})^2/l_d^2$, l_d^2 is an individual lengthscale for each input dimension and σ_n and σ_f are the noise and the signal variances.
In every iteration BO chooses the next query point $\mathbf{x}'$ according to a surrogate function, also called activation surface

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, \alpha(\mathbf{x})\,, \tag{2.12}$$

rather than optimizing the response surface $\tilde{f}$ directly. Different acquisition functions exist [48] that focus on various criteria such as *improvement* (probability of improvement and expected improvement), *optimism* (upper confidence bound) or *information* (Thompson sampling and entropy search) to name just a few. All of them aim at balancing exploration and exploitation to maximize sample efficiency by taking advantage of the full posterior distribution from Equation 2.8

$$\alpha(\mathbf{x}) = \mathbb{E}_{p(\tilde{f}|\mathcal{D},\mathbf{x})}[U(\mathbf{x}, \tilde{f}(\mathbf{x}))]\,, \tag{2.13}$$

where $U(\mathbf{x}, \tilde{f}(\mathbf{x}))$ defines the various quality criteria mentioned above. In particular, we incorporate *expected improvement* (EI)

$$U(\mathbf{x}, \tilde{f}(\mathbf{x})) = \max(0, \tilde{f}' - \tilde{f}(\mathbf{x}))\,, \tag{2.14}$$

where $\tilde{f}'$ is the minimal value of $\tilde{f}$ observed so far. In the context of control parameter tuning, the inputs $\mathbf{x}$ correspond to the control parameter $\boldsymbol{\theta}$ and function evaluations $\tilde{f}(\mathbf{x})$ (targets $\mathbf{y}$ to the GP) are measurements of the control performance $\mathcal{L}$ (we define both in Section 2.3.4). For a comparison of BO approaches to robotics see [49].

2.3.4 Automatic Tuning of PID Parameter for Antagonistic PAMs

It is desirable to automatically tune control parameters directly on the real hardware. A significant concern is the possibility to cause damage to the robot or surrounding objects. Poorly chosen control parameter can cause too fast motions that cannot be decelerated in time. The cause can be fast changing control signal, such as step signals, or instabilities.

On the other hand, high-acceleration robotics tasks require automatic tuning as fast motions are inherently harder to control while being even more susceptible to produce damages. Using our system, we can both generate highly accelerated movements and assure that such motions do not cause damage to the system as described in Section 2.3.1. Consequently, learning control algorithms can experience such explosive motions and incorporate this information rather than avoiding it.
To illustrate this point, we aim at automatically tuning the PID control framework from Section 2.3.1 using the BO approach described in Section 2.3.3. We do so with no further safety considerations than to assume predefined pressure ranges $\mathbf{P}^{\text{lim}} \in \mathcal{R}^{2M\times 2}$ and $[\mathbf{P}^{\text{lim}}]_m = (p_m^{\max}, p_m^{\min})$ that assure the robot to not hit its base. Additionally, we set limits on the parameters $\boldsymbol{\theta}^{\text{lim}}$ that ensure that the maximal instabilities stay bounded (see Section 2.3.1). The system still reaches high velocities and accelerations within these ranges as described in Section 2.4.2.

Algorithm 1 Bayesian Optimization Parameter Tuning of a PID with feedforward compensation and co-contraction

procedure BOPARATUNING($N, \boldsymbol{\theta}^{\text{man}}, \boldsymbol{\theta}^{\text{lim}}, \mathbf{w}, \boldsymbol{\tau}^{\text{des}}$)
  for $i_{\text{dof}} = 1 \dots M$ **do**
    $\mathcal{D} \leftarrow \emptyset$
    $\boldsymbol{\theta}_{i_{\text{dof}}} \leftarrow$ uniformrand()
    for $i_{\text{other}} \neq i_{\text{dof}}$ **do** *// all not current DoFs*
      if $i_{\text{other}} > i_{\text{dof}}$ **then**
        $\boldsymbol{\theta}_{i_{\text{other}}} \leftarrow \boldsymbol{\theta}^{\text{man}}_{i_{\text{other}}}$
      else
        $\boldsymbol{\theta}_{i_{\text{other}}} \leftarrow \boldsymbol{\theta}^{\text{opt}}_{i_{\text{other}}}$
      end if
    end for
    for $i_{\text{it}} = 1 \dots N_{\text{it}}$ **do**
      $\boldsymbol{\tau} \leftarrow$ track($\boldsymbol{\theta}_{i_{\text{dof}}}, \boldsymbol{\tau}^{\text{des}}$)
      $\mathcal{L}_{\text{pos}} \leftarrow (\mathbf{q} - \mathbf{q}^{\text{des}})^T (\mathbf{q} - \mathbf{q}^{\text{des}})$
      $\mathcal{L}_{\text{vel}} \leftarrow (\dot{\mathbf{q}} - \dot{\mathbf{q}}^{\text{des}})^T (\dot{\mathbf{q}} - \dot{\mathbf{q}}^{\text{des}})$
      $\mathcal{L} \leftarrow w_{\text{pos}} \mathcal{L}_{\text{pos}} + w_{\text{vel}} \mathcal{L}_{\text{vel}} + w_{\text{acc}} \mathcal{L}_{\text{act}}(\mathbf{p})$
      $\mathcal{D} \leftarrow [\mathcal{D}, (\boldsymbol{\theta}_{i_{\text{dof}}}, \mathcal{L})]$
      $\boldsymbol{\theta}_{i_{\text{dof}}} \leftarrow \text{BO}(\mathcal{D}, \boldsymbol{\theta}^{\text{lim}})$
    end for
    $\boldsymbol{\theta}^{\text{opt}}_{i_{\text{dof}}} \leftarrow \boldsymbol{\theta}_{i_{\text{dof}}}$
  end for
end procedure

A straightforward approach is to optimize the feedback $\mathbf{k}_i^{\text{fb}} = [k_i^{\text{P}}, k_i^{\text{D}}, k_i^{\text{I}}]$ and feedforward terms $\mathbf{k}_i^{\text{ff}} = [k_i^{\text{pos}}, k_i^{\text{vel}}, k_i^{\text{acc}}]$ for each DoF i. Additionally, we tune the co-contraction parameter p_0 from Equation 2.6. It fundamentally changes how pressures are assigned based on the control signal and hence influences the system's characteristics. Hence, we optimize $\boldsymbol{\theta}_i = [\mathbf{k}_i^{\text{fb}}, \mathbf{k}_i^{\text{ff}}, p_{0,i}]$ for each DoF i separately. We employ our adapted co-contraction approach from Equation 2.6 as it enables p_0 to be part of the tuning (the input range changes using Equation 2.5 but not using Equation 2.6). While optimizing one of the DoFs, the others are either controlled by the previously optimized parameters $\boldsymbol{\theta}_i^{\text{opt}}$ or by manually tuned parameters $\boldsymbol{\theta}_i^{\text{man}}$ that are depicted in Figure 2.10. The tracking using $\boldsymbol{\theta}^{\text{man}}$ are similar to the tacking with $\boldsymbol{\theta}^{\text{opt}}$. Hence, the influence of the other DoFs on the currently tuned DoF stays approximately constant throughout the optimization procedure.
Control performance is hard to measure with a scalar quantity and is inherently multi-objective. Taking inspiration from the LQR framework, we define the losses on position control error

$$\mathcal{L}_{\text{pos}} = (\mathbf{q} - \mathbf{q}^{\text{des}})^T (\mathbf{q} - \mathbf{q}^{\text{des}}) \,, \tag{2.15}$$

and velocity control error

$$\mathcal{L}_{\text{vel}} = (\dot{\mathbf{q}} - \dot{\mathbf{q}}^{\text{des}})^T (\dot{\mathbf{q}} - \dot{\mathbf{q}}^{\text{des}}) \,, \tag{2.16}$$

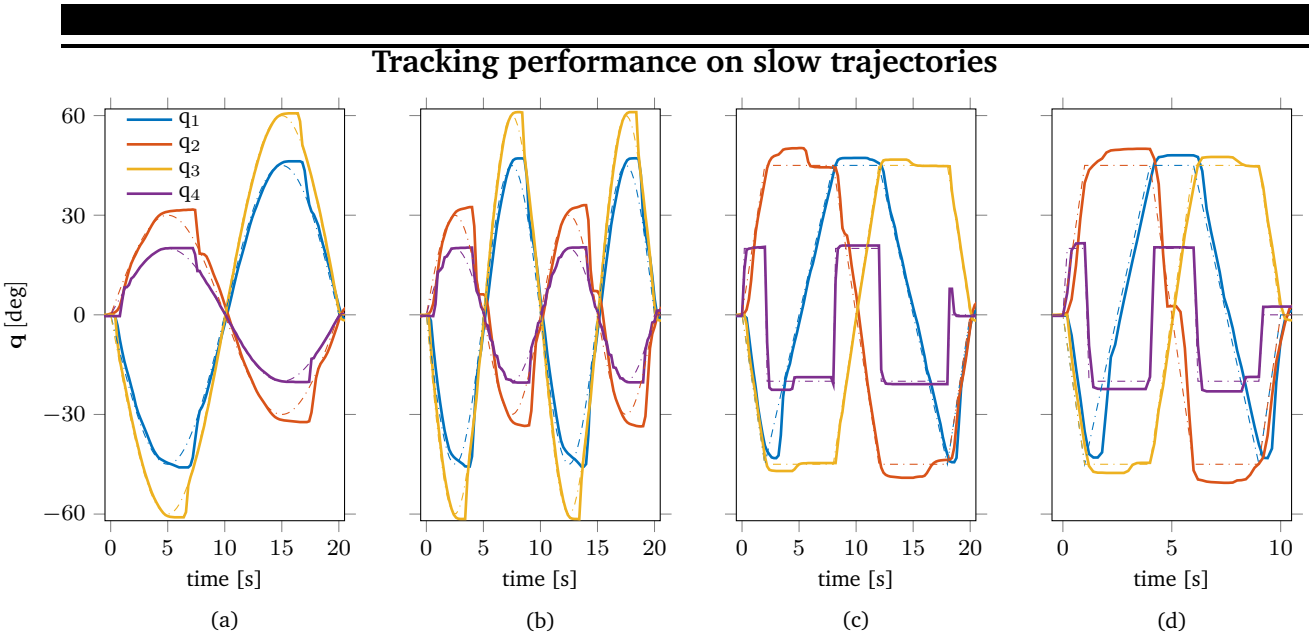


Figure 2.7: The tracking performance shows satisfactory results using a manually-tuned PID controller. For rapid changes in reference signals, some overshoots are visible that cannot be compensated. For smooth changes as in (b) the trajectory is tracked sufficiently well indicating that our construction considerations eased the control of the robot. (a) Sinusoidal reference with f=0.05 Hz. (b) Sinusoidal reference with f=0.1 Hz. (c) Truncated ramp reference for DoF one to three and rectangular reference for DoF four. (d) Same reference as in (c) but twice as fast.

over a given desired trajectory $\boldsymbol{\tau}^{\text{des}} = [\mathbf{q}^{\text{des}}, \dot{\mathbf{q}}^{\text{des}}] \in \mathbb{R}^{N_t \times 2}$ which is different from $\mathbf{q}_t^{\text{des}}, \dot{\mathbf{q}}_t^{\text{des}} \in \mathbb{R}^{\text{M}}$ from Equation 2.4 that indicate joint angles and joint velocities of all DoFs at time t. The goal is also to allow any curvature of the pressures p_a and p_b of PAMs a and b of an antagonistic pair (also step signals) but keep them inside predefined pressure ranges $\mathbf{P}^{\text{lim}}$ over time. This property can be encoded by keeping the control signal u within $[-1, 1]$ using Equation 2.6. Hence, we additionally generate an action loss

$$\mathcal{L}_{\text{act}} = \begin{cases} 0 & \text{if } -1 \leq u \leq 1 \\ |u| - 1 & \text{otherwise} \end{cases}, \tag{2.17}$$

if u is out of its allowed range $[-1, 1]$. We scalarize to a single loss by computing a weighted sum

$$\mathcal{L} = \sum_{k=\{\text{pos,vel,act}\}} w_k \mathcal{L}_k \,, \tag{2.18}$$

which enables us to reuse samples of the losses by saving them separately and reweighting with different $\mathbf{w} = [w_{\text{pos}}, w_{\text{vel}}, w_{\text{act}}]$. A pseudocode of our BO approach is represented in Algorithm 1. In multi-objective optimization, one single set of parameters does not optimize all objectives at the same time, either due to the optimization being stopped early or the objectives contradict each other. The two losses $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{vel}}$ that we use here (omitting the action loss $\mathcal{L}_{\text{act}}$) contradict each other as the linear controller from Equation 2.4 is not capable of achieving perfect tracking although theoretically $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{vel}}$ are zero once one of the objectives is zero. Expressing optimality in the multi-objective case is done by

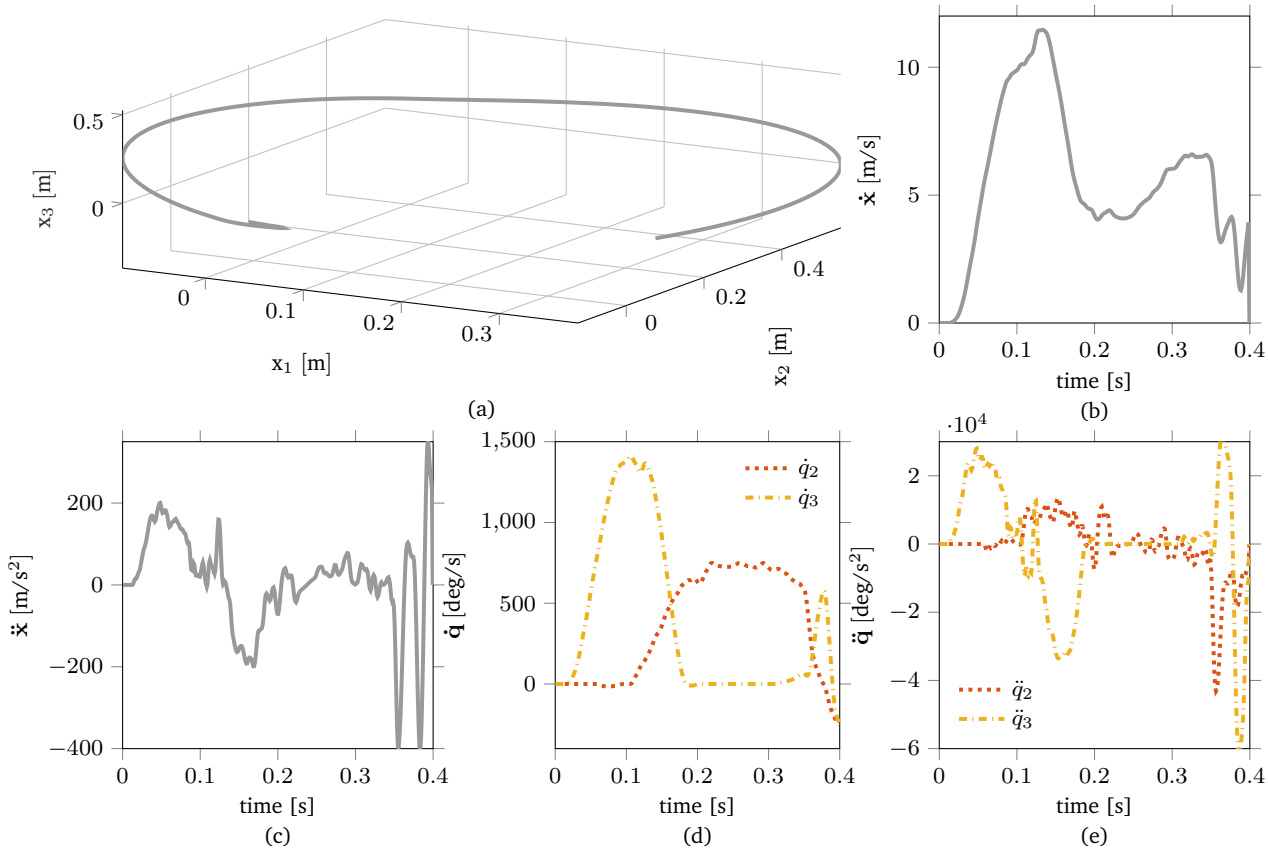


Figure 2.8: High velocity and acceleration profiles in task and joint space. DoF 2 and 3 were actuated with the maximum pressure moving in between the joint limits. (a) Trajectory of the end-effector in task space. (b) Velocity profile along the trajectory in (a). Maximum value is 12m/s. (c) Acceleration profile along the trajectory in (a). The maximum value reaches up to 200 m/s^2. (d) Angular velocity profile for both swivel DoF. DoF 3 is faster as it has to accelerate less weight than DoF 2. The maximum value of about 1500 deg/s is reached with DoF 3. (e) Angular accelerations show a maximum of approximately 28000 deg/s^2

(a) time $= 1\,\mathrm{s}$ (b) time $= 1\,\mathrm{s} \ldots 3\,\mathrm{s}$ (c) time $= 3\,\mathrm{s} \ldots 4\,\mathrm{s}$ (d) time $= 4\,\mathrm{s} \ldots 5\,\mathrm{s}$

Figure 2.9: Images extracted from a video showing the trajectory tracked in Figure 2.10. The images represent distinctive phases of the learned motion: (a) zero position, (b) move to start position, (c) hitting motion and (d) move back to zero position.

calculating the Pareto front (PF). This set consists of points of non-dominated parameters where parameters $\boldsymbol{\theta}_1$ dominate parameters $\boldsymbol{\theta}_2$ if

$$\boldsymbol{\theta}_1 \succ \boldsymbol{\theta}_2 \begin{cases} \forall\, i = [1, N] : \mathcal{L}_i(\boldsymbol{\theta}_1) \leq \mathcal{L}_i(\boldsymbol{\theta}_2) \\ \exists\, j = [1, N] : \mathcal{L}_j(\boldsymbol{\theta}_1) < \mathcal{L}_j(\boldsymbol{\theta}_2)\,, \end{cases} \tag{2.19}$$

which, in other words, means that $\boldsymbol{\theta}_1$ is strictly better in at least one objective compared to $\boldsymbol{\theta}_2$ and not worse in all other objectives.

2.4 Experiments and Evaluations

Having derived our approach to tune control parameters for our developed system automatically, we now perform experiments to demonstrate its feasibility. First, we show that our construction considerations make it possible to track slow trajectories with PIDs only. In the second experiment, we demonstrate high velocity and acceleration motions to underline the arm's ability to be used for hitting and generate catapult-like motions while avoiding damages at such paces. In the last experiment, both preceding experiments are combined by learning control (PID with feedforward terms and co-contraction) directly on the real hardware without additional safety considerations on a fast and hitting-like trajectory.

2.4.1 Control of Slow Movements

This experiment highlights the arm's low controlling demands by showing that adequate tracking performance is possible on slow trajectories using linear controllers only. Therefore, we track all four DoFs simultaneously for two kinds of reference signals, as can be seen in Figure 2.7. In Figure 2.7c and Figure 2.7d a truncated triangular signal was tracked for $10\,\mathrm{s}$ and $20\,\mathrm{s}$, respectively. The controller from Equation 2.3 has been used and the co-contraction parameter from Equation 2.6 is $p_0 = 0$. All graphs show that for rapidly changing references, tracking becomes inaccurate. This deficiency is caused by the PIDs assuming a linear system while PAMs are inherently nonlinear. Additionally, for abrupt corrections, in the first moments, the change of pressure in the PAM does not affect the joint angle in case the PAMs are not co-contracted enough [11]. For severe cases, this forbearance is followed by a too strong correction as can be seen for DoF two in Figure 2.7a

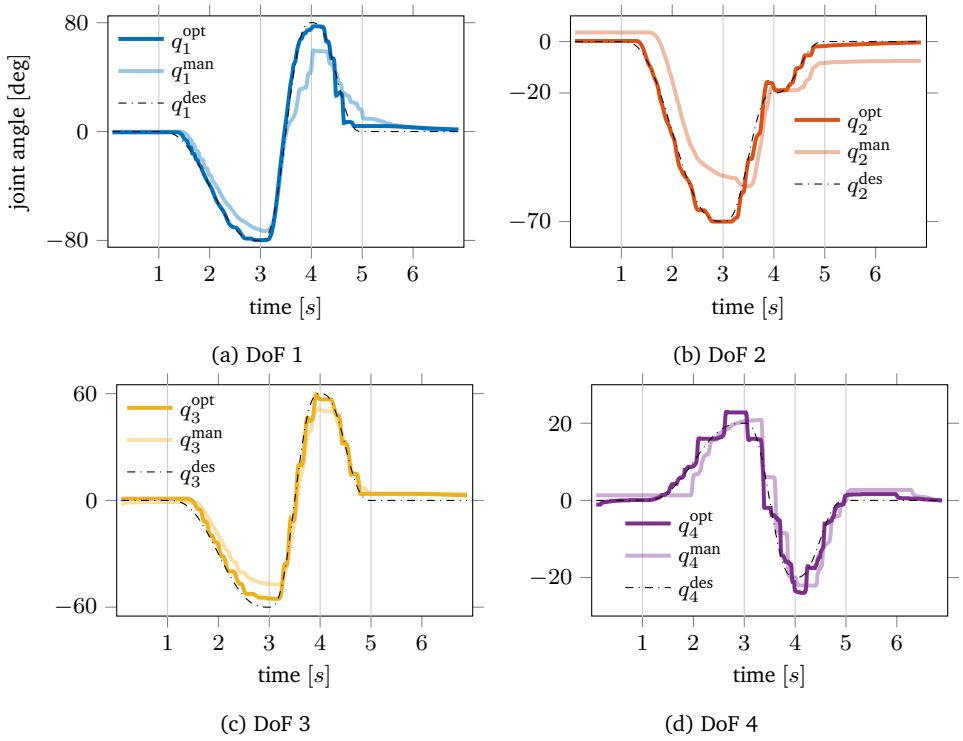


Figure 2.10: Tracking performance for all degrees of freedom after optimization using Bayesian optimization (indicated by the subscript 'opt') and after manual tuning by an expert on the system (indicated by the subscript 'man'). The trajectory tracked resembles an example of a fast hitting motion between $t = [3\,\mathrm{s}, 4\,\mathrm{s}]$. It is composed of a slow motion towards the start position ($1\,\mathrm{s}...3\,\mathrm{s}$) followed by a fast hitting motion ($3\,\mathrm{s}...4\,\mathrm{s}$) and a motion back to the zero position ($4\,\mathrm{s}...5\,\mathrm{s}$) (see Figure 4.5). The manual parameters are used to track the currently not optimized DoFs in Algorithm 1 as $\boldsymbol{\theta}^{\mathrm{man}}$. It is apparent that tracking quality for DoFs three and four is more impaired as for DoFs one and two due to higher friction as the cables are guided through the arm. Additionally, the PID with feedforward compensation assume a linear system, hence, BO optimizes towards higher gains as the system is heavily nonlinear.

and c for the middle part of the graph. This DoF drives the most mass and hence is harder to control precisely compared to the other DoFs. Sub-figures 2.7b and d show tracked sinusoidal references with 0.05 and $0.1\,\mathrm{Hz}$. Here the same issues occur for rapid changes of the reference. However, for smooth changes, the reference can be followed with some small delay with all DoF.

2.4.2 Generation of Ballistic Movements

High accelerations are necessary to reach high velocities on a short distance to enable a versatile bouquet of possible trajectories and fast reactions. Our system can generate high velocities and accelerations due to the strength of the PAMs used while being robust due to the antagonistic muscle configuration. This property is critical for the exploration of fast hitting motions using learning control methods. In this experiment, we show that the system can sustain the fastest possible motion that can be generated with our system using the rotational DoFs two and three. The respective minimum pressure was set to one of the PAMs of each muscle pair while the maximum pressure was assigned to the antagonist. The subsequent switching from maximum to minimum and vice versa generated a fast trajectory at the end-effector, as shown in Figure 2.8a. Note that this step set signal generates the fastest movement at the end-effector that a closed-loop controller could have determined without instabilities. We did not find any other set signal that moved the arm that close to its joint limits and generated such high peak velocities and accelerations. The task space $\mathbf{x} = [x_1, x_2, x_3]^T$ has been determined from the joint space coordinates $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ for each data-point using the forward kinematics equations $\mathbf{x} = T_x^q(\mathbf{q})$. The forward kinematics equations can be derived from Figure 2.4b. We do not consider the orientation of the end-effector here. The resulting velocity and acceleration profiles, depicted in Figure 2.8b and c, show at their respective maxima approximately $12\,\mathrm{m\,s^{-1}}$ and $200\,\mathrm{m\,s^{-2}}$. As a comparison, the fast Barrett Wam arm used for table tennis in [36], can generate peak velocities of $3\,\mathrm{m\,s^{-1}}$ and peak accelerations of $20\,\mathrm{m\,s^{-2}}$. The resulting angular velocities in DoF three reaches up to $1400\,^\circ\mathrm{s}^{-1}$ and angular acceleration of $28\,000\,^\circ\mathrm{s}^{-2}$.

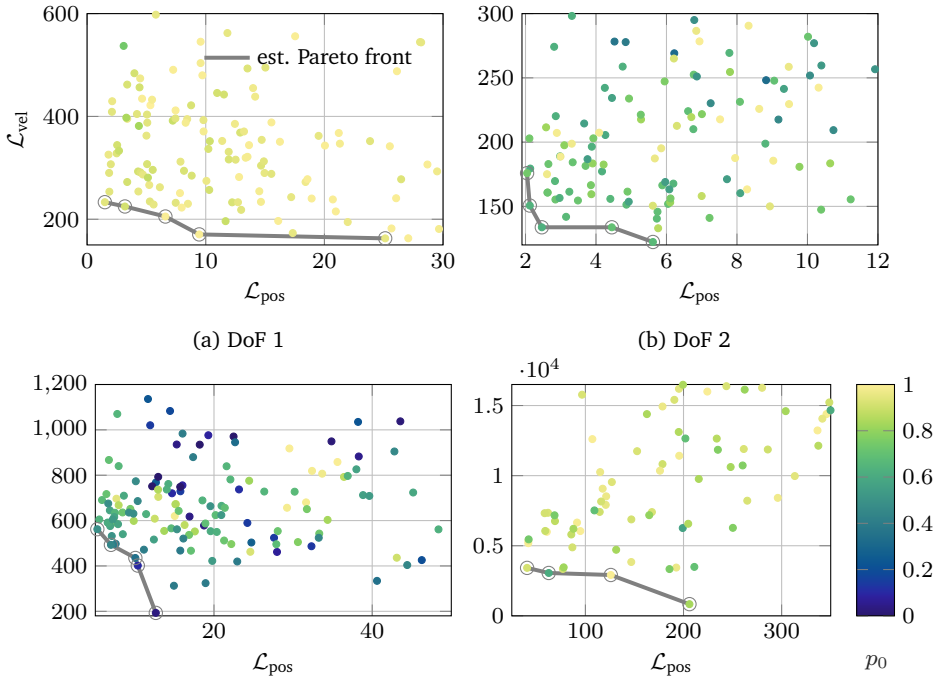
2.4.3 Bayesian Optimization of Controller Parameters

Having demonstrated that the robot arm can be controlled using simple PIDs for slow trajectories and that the system sustains fast hitting motions, the natural next goal is to learn to control fast trajectories (see Figure 4.5). At higher speeds, tuning control parameters lead to potentially dangerous configurations on traditional motor-driven systems that we can partially avoid using antagonistic actuation, as discussed in Section 2.3.1. We tune seven parameters

$$\boldsymbol{\theta}_i = [k_i^{\mathrm{P}}, k_i^{\mathrm{D}}, k_i^{\mathrm{I}}, k_i^{\mathrm{pos}}, k_i^{\mathrm{vel}}, k_i^{\mathrm{acc}}, p_{0,i}] \tag{2.20}$$

for each of the four DoFs $i = 1 \ldots 4$ (28 parameters in total) where the additional feedforward components from Equation 2.4 improve control on fast trajectories.
In addition to the feedback and feedforward components' parameter, we also optimize the co-contraction parameter p_0 from Equation 2.6. Too much co-contraction increases

Similar co-contraction levels appear close to the PF in the objective space



(a) DoF 1 (b) DoF 2

(c) DoF 3 (d) DoF 4

Figure 2.11: Objective space that spreads the velocity $\mathcal{L}_{\text{vel}}$ and position objective $\mathcal{L}_{\text{pos}}$ from Equation 2.16 and Equation 2.15for all four DoFs where each point represents one tracking instance of the trajectory from Figure 2.10. The color indicates the value of co-contraction parameter p_0. Note that the figures are zoomed in to illustrate the estimated Pareto front (PF), hence, differently colored points lie outside this area. It is apparent that points with similar p_0 appear close to the estimated PF instead of being diverse. Substantially different colors are almost not present (except of for DoF three) as they lie outside the zoomed area. The dominant co-contraction range close to the PF is $p_{0,1} = 0.9 \dots 1$ for DoF one (a), $p_{0,2} = 0.5 \dots 0.7$ for DoF two (b), $p_{0,3} = 0.2 \dots 0.5$ for DoF three (c) and $p_{0,4} = 0.6 \dots 0.9$ for DoF four (d).

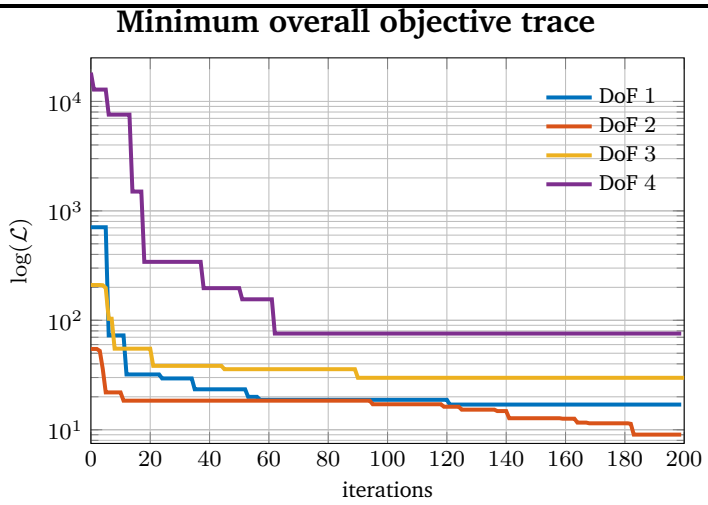


Figure 2.12: Minimum overall objective trace over 200 iterations for each DoF. All DoFs start at different initial values as the parameter for the first iteration are drawn randomly. After approximately 60 iterations the performance has improved by at least one order of magnitude. The main contributer is that our approach uses additionally potentially dangerous data points which would break traditional motor-driven systems.

the friction within the tendon-driven system, whereas too low p_0 complicates control as the tendons are not stretched for some configurations. Hence, we aim to answer whether the stiffness in a joint - using the co-contraction p_0 as a proxy - influences the tracking significantly. Optimizing 28 parameters in total renders manual tuning impossible. Figure 2.10 depicts the tracking performance that is achieved after two hours of manual tuning (q_i^{man}) by an expert on the system. Especially troublesome is the interdependence of the DoFs. Tuning one DoF to a satisfactory performance often impairs the tracking of the other DoFs. Hence, we incorporate the BO tuning scheme from Algorithm 1 (described in Section 2.3.4) to tune one DoF at a time and apply either manual parameters $\boldsymbol{\theta}_i^{\text{man}}$ or the optimized parameters $\boldsymbol{\theta}_i^{\text{opt}}$ in case they were already found to the other DoFs i. We use the position $\mathcal{L}_{\text{pos}}$ and velocity losses $\mathcal{L}_{\text{vel}}$ from Equation 2.15 and Equation 2.16 respectively as performance measure. Although we also employ the action loss $\mathcal{L}_{\text{act}}$ (Equation 2.17), it merely acts as a regularizer to allow all curvatures of the action signal within the allowed range $[-1, 1]$ while penalizing exceeding these limits. Figure 2.10 illustrates the tracking performance found (q_i^{opt}) by applying Algorithm 1 and Table 2.2 compares the mean squared error between the corresponding tracking performances. The algorithm converges quickly after 200 iterations for each DoF, as shown in Figure 2.12. To the best of our knowledge, this is the fastest tracked trajectory with a four DoF PAM-based robot (see Figure 4.5 for an illustration of the trajectory). We think that the main advantage gained is that our BO tuning procedure additionally learns from data points that would break traditional robotic systems. As a result, our approach pioneers the direct application of Bayesian optimization on a real system on a task that potentially breaks the system. Note that the parameters we found are only valid in the vicinity of the desired trajectory we chose. The reason is that the control architecture is inherently linear and hence the same parameters are generally valid for linear systems.

The amount of co-contraction p_0 is an essential property of PAM-based systems. An infinite set of pressures in one PAM pair leads to the same joint angle; the co-contraction discriminates this infinite set. Also, it correlates with the stiffness in the DoF. An interest-

ing question is whether the right choice of co-contraction improves control performance. To answer this question, we study the connection between loss and co-contraction. From all the points for each DoF, we calculated the estimated Pareto front (PF) using Equation 2.19 and colored the data depending on the value of the co-contraction parameter p_0. Figure 2.11 shows the objective space for each DoF, where one point represents one tracking instance. The figure spans the position $\mathcal{L}_{\text{pos}}$ as well as the velocity objective $\mathcal{L}_{\text{vel}}$ from Equation 2.15 and Equation 2.16. We left out the regularizing action objective $\mathcal{L}_{\text{act}}$. Not all data points are visible as the figures are zoomed to illustrate the PF more clearly. It is apparent that close to the PF, the values of p_0 are similar. In particular, for DoF one, two, and four, significantly different colors do not appear in the figure at all as they lie outside the zoomed-in area. The absence of a whole range of co-contraction levels is a strong hint that by choosing p_0 conveniently, a linear controller can better control our generally nonlinear system.

2.5 Conclusion

Generating high accelerations with robotic hardware as seen in human arm flick motions while maintaining safety during the learning process are desirable properties for modern robots [4]. In this chapter, we built a lightweight arm actuated by PAMs to try to address this issue. Our robot avoids fundamental problems of previous PAM-driven arms, such as unnecessary directional change of cables and PAMs that are bent around or touch the structure or other PAMs. We experimentally show that our system eases control by tracking a slow trajectory sufficiently while incorporating only simple and manually tuned linear PID controllers. Execution of ballistic movements illustrates that our arm surpasses the peak task space velocity and acceleration of a Barrett Wam arm by a factor of 4x and 10x, respectively. Experiments also show that our system can sustain the large forces generated by high accelerations and antagonistic actuation. This property allows machine learning algorithms to explore in high velocity regimes without taking safety into account. To demonstrate this ability, we automatically tuned PID controllers with additional feedforward components using BO to learn to track a fast trajectory. Predefined pressure and parameter limits were sufficient to avoid damaging the robot itself and prevent unbounded instabilities during training. We did not find any other reported publication that tracked faster trajectories with a comparable PAM-driven system. Data points collected during training indicate that trials close to the estimated PF own similar co-contraction levels. In this manner, we empirically illustrate that the choice of the co-contraction level has a significant influence on the performance at trajectory tracking tasks. From these results follows that softly actuated robots offer robustness, which enables the application of powerful algorithms that, in turn, enable to perform complex tasks at a high level of performance. Interesting future directions are to extend the system to six DoFs to allow to reach arbitrary positions and orientations with the end-effector. This extension enables us to perform more complex tasks that require a higher number of DoFs, such as table tennis (requires at least five DoFs). We are also curious to learn low-level control (in muscle space), for instance, using reinforcement learning (RL), to perform more complex tasks that involve fast motions. By allowing RL to explore in fast regimes, RL might be able to learn well-performing policies. These policies have the chance to be less conservative as

safety wrt robot intactness is handled inherently. Another interesting direction is to understand the extent to which pretrained policies in simulation can be used directly on the real system for high-acceleration tasks. Lastly, it is intriguing to extend the research on how to leverage the overactuation derived from the co-contraction. This inherent property is essential when tuning controllers for trajectory tracking, as shown in this chapter, but might pose more challenging control demands due to increased state dimensionality, etc.

Table 2.1: A collection of pneumatic based robotic arm-like systems, listed next to the number of DoFs in joint space and the fastest and most complex tracked trajectory if known.

Year	Publication	# DoF	Fastest and most complex trajectory tracked
2018	Driess et al. [13]	2 (5 PAMs)	Reaching motions
2016	Das et al. [14]	2	Step signal to both DoFs
2014	Rezoug et al. [15]	7	Sinusoidal reference with $f = 1\,\mathrm{Hz}$ for one DoF
2012	Hartmann et al. [16]	7	Sinusoidal reference in task space (x: $f = 1\,\mathrm{Hz}$, y: $f = 2\,\mathrm{Hz}$, z not tracked)
2012	Ikemoto et al. [17]	7 (17 PAMs)	Human taught reference (similar to sinusoidal) periodic with $f{\sim}0.33\,\mathrm{Hz}$
2009	Ahn and Ahn [18]	2	Triangular reference with $f = 0.05\,\mathrm{Hz}$
2009	Shin et al. [19]	1 (4 PAMs)	Sinusoidal reference with $f = 6\,\mathrm{Hz}$ for one DoF
2009	Van Damme et al. [20]	2 (4 PAMs)	Sinusoidal reference with $f = 0.33\,\mathrm{Hz}$ for both DoFs
2007	Festo Airic's arm [21]	7 (30 PAMs)	N/A
2006	Thanh and Ahn [22]	2	Circular with $f = 0.2\,\mathrm{Hz}$ using both DoFs
2005	Hildebrandt et al. [23]	2	Step and sinusoidal reference with $f = 0.5\,\mathrm{Hz}$
2005	Tondu et al. [24]	7	N/A
2004	Boblan et al. [25]	7 in arm	N/A
2000	Tondu and Lopez [26]	2	Independent sinusoidal activation of each DoF with $f = 0.1\,\mathrm{Hz}$
1998	Caldwell et al. [27]	7	Response of shoulder joint to $90°$ step reference
1995	Caldwell et al. [28]	7	Response to a square wave reference input ($f = 0.2\,\mathrm{Hz}$, 1 DoF)

Table 2.2: Root mean squared error (RMSE) comparison of tracking objectives for manually tuned parameters (MANUAL, $\boldsymbol{\theta}^{\mathrm{man}}$) from Figure 2.10 and tracking with parameters found with our Bayesian optimization approach (Algorithm 1) from Figure 2.10 (OPTIMIZED, $\boldsymbol{\theta}^{\mathrm{opt}}$).

# DoF	RMSE manual	RMSE optimized
1	12.08	3.29
2	11.06	2.00
3	7.22	3.74
4	3.47	2.11

3 Control of Musculoskeletal Systems using Learned Dynamics Models

In the previous chapter, we built a lightweight robotic arm with eight PAMs. We illustrated its robustness at fast movements due to the antagonistic actuation which allows to explore dynamic tasks as required for learning control approaches. We showed that by automatically tuning the feedforward, feedback and co-contraction terms, it is possible to achieve precise trajectory tracking of a fast and hitting-like motion. Controlling such musculoskeletal systems, however, still remains a challenging task due to nonlinearities, hysteresis effects,massive actuator delay and unobservable dependencies . Despite such difficulties, muscular systems offer many beneficial properties to achieve human comparable performance in uncertain and fast-changing tasks in addition to the benefits mentioned in the previous chapter. For example, muscles are backdrivable and provide variable stiffness while offering high forces to reach high accelerations. In addition, the *embodied intelligence* deriving from the compliance might reduce the control demands for specific tasks.

In order to use these properties, it is important to address the problem of how to improve accurate control of musculoskeletal robots using learned models. Analytical models of such robots have been proposed for multiple decades. We thing that models learned from data have a higher chance to exceed existing performance than an additional effort of classical modeling of musculoskeletal systems. In particular, we propose to learn probabilistic forward dynamics models using Gaussian processes, and, subsequently, to employ these models for control. However, Gaussian processes dynamics models cannot be set-up for our musculoskeletal robot as for traditional motor-driven robots because of unclear state composition etc. We hence empirically study and discuss in details how to tune these approaches to complex musculoskeletal robots and its specific challenges. Moreover, we show that our model can be used to accurately control a real four DoF robot arm driven by eight pneumatic muscles for a trajectory tracking task while considering only 1-step ahead predictions of the forward model. Key to the good tracking performance is regularization with variances of the Gaussian process.

3.1 Introduction

Most dynamic activities that appear straightforward for humans, such as walking, grasping or ball games, are still fundamental challenges in robotics. Uncertainty, required fast reactions and necessary high accelerations – without damaging the system and environment – still pose big hurdles. Despite the existence of algorithms that outperform humans in non-robotics tasks [1], the transfer to super-human performance to robots in dynamic tasks has not been shown yet.

Using muscular robots can be a way to achieve human-level performance in robotics. In this work, we try to leverage antagonistic pairs of pneumatic artificial muscles (PAMs) instead of traditional motors. PAMs are the nearest replica of skeletal muscles available as robotics hardware and share many desired properties [43]. First, PAMs are backdrivable, thus, damages at low velocity impacts with humans, external objects and the robot itself are reduced although not completely prevented. On the other hand, changing co-contraction levels adjust the compliance in the antagonistic pair, offering flexibility if the task requires it. Second, high accelerations, provided by PAMs, enable the robot to reach desired states in less time and allow for fast flick-movements due to energy storage as observed in fast human arm movements. Third, learning dynamic tasks, e.g. table tennis, is more feasible with antagonistic actuation as damage due to exploration at higher velocities can be minimized [4].

Fourth, it has been shown that the demands on the control algorithm are reduced for tasks where contact with external objects is required, e.g. opening a door [50, 51]. Also muscular actuation assures gait stability despite of the presence of unmeasured disturbances [52], a desirable property that might cope with uncertainties of dynamic tasks. These insights illustrate what is known as embodied intelligence and may – in combination with a learning control approaches – pave the way to human-level movements.

Yet, muscular robots are not widely used. Many of the issues with pneumatic muscles ultimately derive from the lack of good dynamics models that generally describe the relationship between the action $\mathbf{a}$ taken in state $\mathbf{s}$ and the successor state $\mathbf{s}'$. Severely non-linear behavior, unobservable dependencies like temperature, wear-and-tear effects and hysteresis [43] renders modeling of PAM systems considerably challenging.

Another reason why muscular robots are scarcely in use is overactuation. An infinite set of air pressure combinations in an antagonistic PAM pair lead to the same joint angle but with different compliance levels. Compliance $c = \delta q / \delta F_{\text{ext}}$ or its inverse the stiffness k represents the external force F_{ext} applied to the joint required to cause a change of δq in the joint angle. Overactuation is critical for two reasons. First, overactuation principally rules out acquiring inverse models $\mathbf{a} = f(\mathbf{s}, \mathbf{s}')$ of musculoskeletal systems by traditional regression. Inverse dynamics models are only unique for traditionally actuated robots and have to be approximated locally for over- and under-actuated systems [53]. The second reason is, that overactuation expands the state-action space, thus, learning such dynamics models requires more data or additional assumptions about the underlying mechanisms. Using forward models $\mathbf{s}' = f(\mathbf{s}, \mathbf{a})$, however, does not fully solve the issue despite describing the true causal relationship and hence always exist. In interplay with a controller, the actions are based on predictions of the forward dynamics model.

Without any additional constraints, such a controller would find potentially equally optimal outputs and finding optimal actions results in optimization of non-convex loss functions.

The straightforward constraint is minimal energy, hence always choosing the minimal pressures that lead to the minimal control error. Another possibility is to change stiffness or compliance according to the task, e.g., for grasping of a delicate objects. The main idea of our approach is to pose a constraint that eases the use of a learned non-parametric and probabilistic models. In the face of the above-mentioned difficulties of retrieving good models, learning a flexible model purely from data seems to be a promising way. Recent

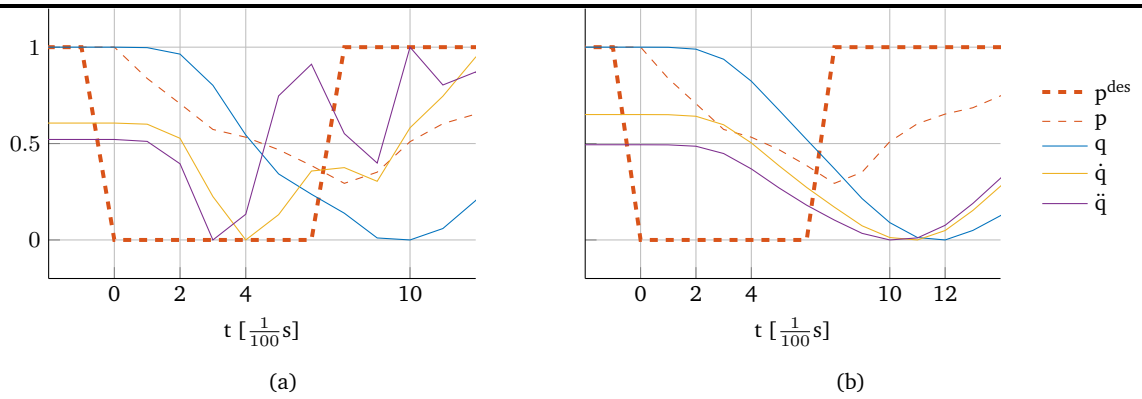


Figure 3.1: System responses to a step in control signal at $t = 0$. All values have been normalized to be in $[0, 1]$. It can bee seen that $\ddot{q}$ reaches its minimum faster than $\dot{q}$ and q. (a) Unfiltered sensor values show a faster response as any filtering adds delay. The first substantial changes occurs at $t = 2$. (b) Filtered values reach their respective minimum slower than in (a). The first substantial changes occurs at $t = 3$, hence 1 time step later than the unfiltered signals.

successes in Gaussian process (GP) dynamic models [54, 55, 56] are especially encouraging to follow this line of research. However, the application of nonparametric models in an online setting requires – among other considerations – the selection of a small set of informative training data points from a stream of highly time-correlated data stream. Hence, it is possible to deviate into unknown state-action regions, especially with time-variant systems like pneumatic muscles. In order to enable nonparametric model learning for antagonistically actuated systems despite higher dimensionality due to overactuation, we use the uncertainty of the GP model as an additional constraint.
The contribution of this chapter is twofold. First, we discuss important aspects of learning models for muscular systems as they are substantially different to traditional robots. The derived key aspects are empirically compared. Second, we show how to incorporate the uncertainty of the GP dynamics model into our control framework so as to make use of the overactuation to stay in the vicinity of the training data.

3.2 Model Learning & Control

Learning flexible and probabilistic forward dynamics models and using them for control is a promising way to achieve better performance with muscular robots. Gaussian process (GP) regression is a non-parametric and probabilistic model that is widely used to represent robot dynamics [53]. Here, we address how to adapt such a GP model for muscular systems and discuss the additional difficulties that arise compared to dynamics modeling for traditionally actuated robots. Following, we show how to use uncertainty estimates of the GP model during control to exploit the overactuation inherent to musculoskeletal systems in order to generate trajectories which stay close the training data.

3.2.1 Traditional Model Learning with Gaussian Processes

Muscular robots are substantially different to motor-driven systems. After giving some background on forward dynamics modeling with GPs of motor-driven robots, we briefly describe the Hill-muscle model to illustrate our adaption of the GP model setup.

Generally, the dynamics of a robot that represents the relationship between joint angles $\mathbf{q}$ and its derivatives $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ to the torque vector τ are modeled by differential equations derived by applying Newtons second law and assuming rigid links

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}\,, \tag{3.1}$$

with $\mathbf{M}$ being the joint space inertia matrix, $\mathbf{C}$ represents the Coriolis and centrifugal forces and $\mathbf{g}$ gravitation. For traditional motor-based robots the torque τ is proportional to the input current. Hence the relationships $(\mathbf{q}, \dot{\mathbf{q}}, \tau) \rightarrow \ddot{\mathbf{q}}$ and $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \rightarrow \tau$ represent forward and inverse dynamics. The state here is clearly defined to be $\mathbf{s} = \{\mathbf{q}, \dot{\mathbf{q}}\}$ and $\ddot{\mathbf{q}}$ can be used as successor state $\mathbf{s}'$ because the actions τ directly influence $\ddot{\mathbf{q}}$ and the state entities can be recovered by differentiation. For muscular robots the state composition is not obvious. The torque $\tau_i = \mathbf{F}_i \times \mathbf{r}_i$ depends on the combined force of all muscles acting on joint i, $\mathbf{F}_i$, as well as the geometry of the joint, specifically the moment arm $\mathbf{r}_i$. For an antagonistic muscle pair with a radial joint, the torque reduces to

$$\tau_i = (F_a - F_b)r_i\,, \tag{3.2}$$

given the forces of both muscles F_a and F_b and the angle independent radius r_i.

Analytical Muscle Model

Many analytical force models for skeletal muscles exist. One of the most widely used is the Hill muscle model, see [57] for a detailed description. It has been shown that the Hill model reflects the properties of PAMs to some extend [32]. The total muscle force

$$F_M = F_{CE} + F_{PEE} = F_{SEE}\,, \tag{3.3}$$

derived from this model is based on an active contractile element F_{CE} that depends on the activation a and a passive parallel and serial elastic element F_{PEE} and F_{SEE} that both change with the muscle length l_M. The active part of the total force

$$F_{CE} = aF_{max}f_L(l_{CE})f_V(v_{CE})\,, \tag{3.4}$$

depends not only on the activation a but also on the force-length f_L and the force-velocity f_V relationship and is parameterized by the maximum isometric force F_{max}. Typically, f_L is bell-shaped whereas a sigmoid-like function constitutes f_V. Equation 3.4 can than be used to form the dynamics of the muscle length

$$\frac{\partial l_{CE}}{\partial t} = f_V^{-1}\left(\frac{F_{SEE} - F_{PEE}}{aF_{max}f_L(l_CE)}\right)\,, \tag{3.5}$$

that takes the activation a and l_{CE} as parameters, resulting from the interaction of τ in Equation 3.1. Often the activation is modeled as a non-instantaneous process based on a neural excitation signal u

$$\frac{\partial a}{\partial t} = c_a(a - u)\,, \tag{3.6}$$

in which c_a is the constant activation and deactivation rate.

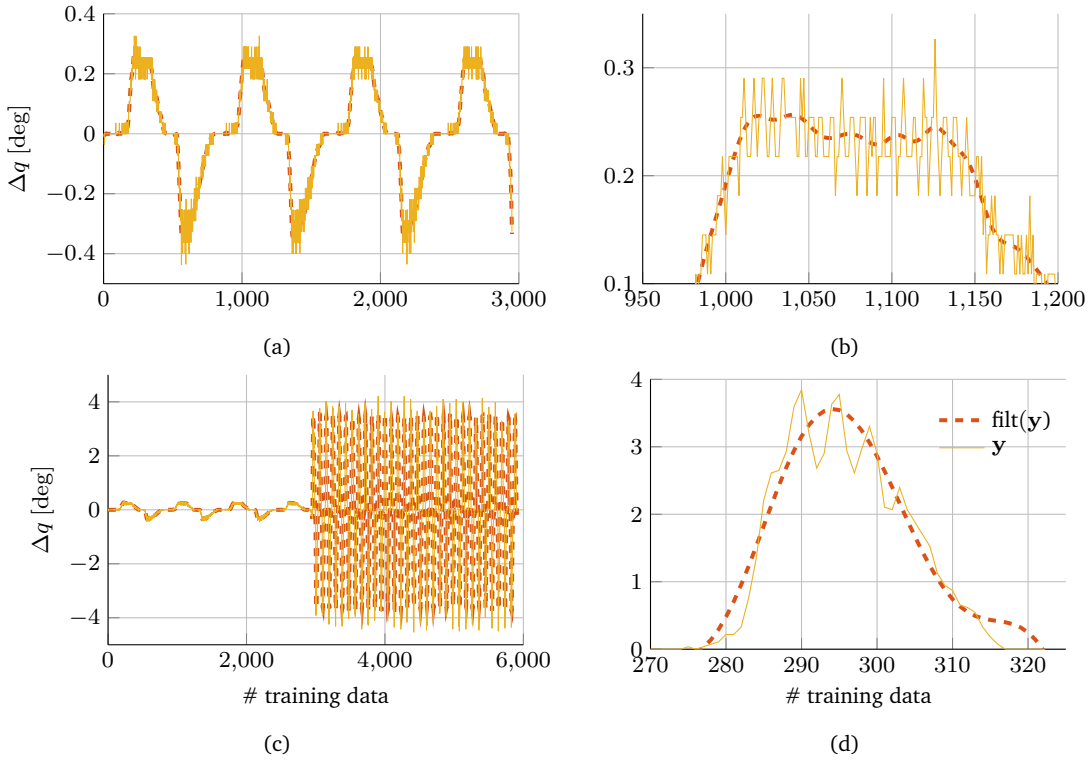


Figure 3.2: Datasets under consideration in this work depicted for illustration. All datasets were created by applying periodic pressure trajectories and recording the resulting sensor values. The target Δq represents the difference to the next state and is depicted in filtered and raw sensor form. A non-causal 10th order butterworth lowpass filter was used for filtering. (a) 'slow' dataset. (b) Closer view of (a) illustrates that the relative angle encoders add considerable noise for slow motions. (c) 'slow' and 'fast' datasets form the 'mixed' dataset. (d) Closer view of fast part of (c). For faster movements the noise smaller. Hence, the 'mixed' dataset includes two types of noise. Faster movements excite higher order dynamics components which need to be expressed by the model.

Gaussian Processes Forward Dynamics Models

In this chapter, we learn a probabilistic Gaussian Process [9] forward dynamics model in discrete-time

$$\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t) + \epsilon\,, \tag{3.7}$$

with $\mathbf{s} \in \mathbb{R}^D$ being the state, $\mathbf{a} \in \mathbb{R}^M$ the control action, and $\epsilon \sim \mathcal{N}(0, \Sigma_n)$ i.i.d. Gaussian measurement noise with a diagonal covariance noise matrix Σ_n. The state transfer function f is modeled by a Gaussian process with a squared exponential kernel and automatic relevance determination (ARD)

$$k(\mathbf{x}_a, \mathbf{x}_b) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_a - \mathbf{x}_b)^T \Lambda^{-1} (\mathbf{x}_a - \mathbf{x}_b)\right), \tag{3.8}$$

having signal variance σ_f^2 and squared lengthscales $\Lambda = \text{diag}(l_1^2, \ldots, l_{D+M}^2)$. The data set $\mathcal{D} = \{X, \mathbf{y}\}$ consist of a design matrix $X \in \mathbb{R}^{N \times D+M}$ with each row being the nth training input $\mathbf{x}_n^T = [\mathbf{s}_n, \mathbf{a}_n]^T$ and $\mathbf{y} \in \mathbb{R}^N$ the target values. Hence, one GP is established for each element of $\mathbf{s}'$. A GP can be seen as a distribution over functions and is queried using the conditional (posterior) probability

$$p(\mathbf{f}|X, \mathbf{y}, \mathbf{x}_*) = \mathcal{N}(\mathbf{k}_*^T \alpha, k_{**} - \mathbf{k}_*^T \tilde{K} \mathbf{k}_*)\,, \tag{3.9}$$

where $[\mathbf{k}_*]_n = k(\mathbf{x}_*, \mathbf{x}_n)$, $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$, $\alpha = \tilde{K}\mathbf{y}$ and $\tilde{K} = (K + \sigma_n)^{-1}$. The hyper-parameters σ_f, σ_n and Λ are optimized by maximizing the marginal likelihood $p(\mathbf{y}|\mathbf{X})$. The GP provides a flexible model by assuming only smoothness of the underlying function f, depending on the choice of the covariance function. On the other hand, a this non-parametric training data set is required to make predictions which needs to be kept and the prediction time complexity rises cubical with its size N $\mathcal{O}(N^3)$. Many approximations schemes exist [58, 59] that are term to future work but are not used here as they rely on approximations.

3.2.2 Model Adaptations to Muscular Systems

Many properties of pneumatic muscular systems render modeling difficult. We mention some of this issues from which we subsequently derive adaptations to the GP forward dynamics model.

Modeling Issues

Modeling of PAMs is still a key problem for control [43]. The reason for this is that analytical models derived from physics, including the Hill muscle model from Equation 3.4, do not sufficiently describe aspects of real PAMs. Unobserved influences like volume change of the muscle while moving, tip effects and temperature as well as hysteresis, nonlinearities require the model to be extraordinary flexible. Additionally, some parameters in analytical muscle models are kept constant for simplicity and avoiding high computational load.

Another important issue with pneumatics is actuator delay. In every system a short time passes between applying control until a reaction is sensed. The time in between sums up all delays in the control loop. For instance, a magnetic field that has to increase in a motor or – as in our case – valves have to open and air pressure has to rise in PAMs. For PAM-actuated robots, the sequence of actuation

$$\mathbf{p}^{\text{des}} \rightarrow \mathbf{u}^{\text{valve}} \rightarrow \text{air flow} \rightarrow \mathbf{p} \rightarrow \tau \rightarrow \ddot{\mathbf{q}} \rightarrow \dot{\mathbf{q}} \rightarrow \mathbf{q}\,,$$

contains more sources of delay compared to motor-driven systems, e.g., mechanical opening and closing of the valve and the generation of air pressure in the muscles. This process is further delayed by the compressibility of air. Figure 3.1 shows an experiment where the joint angles $\mathbf{q}$, velocities $\dot{\mathbf{q}}$ and accelerations $\ddot{\mathbf{q}}$ are recorded in response to a step in desired pressure signal $\mathbf{p}_{\text{des}}$ for unfiltered and filtered case. Unfiltered entities respond faster to the excitation but inhere more noise, especially higher time-derivatives like joint velocities $\dot{\mathbf{q}}$ and accelerations $\ddot{\mathbf{q}}$ because they need to be estimated. Strong noise complicates modeling with GPs as only output noise is assumed.

State Composition

Another consequence from severe actuator delay is that the successor state $\mathbf{s}'$ is dependent on the current state $\mathbf{s}$ and action $\mathbf{a}$ as well as previous states and actions. When formulating a control task as a Markov decision process (MDP, [60]), the forward dynamics or transition function is, however, required to establish a relationship where $\mathbf{s}'$ depends only on $\mathbf{s}$ and $\mathbf{a}$. In an MDP the state has to bear the Markov property

$$p(\mathbf{s}_t|\mathbf{a}_{1:t}, \mathbf{s}_{0:t-1}) = p(\mathbf{s}_t|\mathbf{a}_t, \mathbf{s}_{t-1}). \tag{3.10}$$

This requirement is often violated when the dependence on the past is weak and lead to a model that is sufficiently good. On real systems, dependencies often remain unmeasured due to the lack of sensors or tedious measurement procedures. Estimating the temperature in PAMs is an example which influences the behavior of the rubber material as well as the compressibility of the air. Ideally, the temperature is measured over the complete length of the muscle, on the material itself and the air inside the muscle. This is not feasible and our system is not equipped with temperature sensor. The same line of thinking is true for stiction and friction effects as well as slack of the cables. Nonetheless, the information of such unobserved effects is captured in the transition $(\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbf{s}_{t+1}$ as different successor states will be reached when applying the same action in the same state at different times. Such a problem is described as a partially observable MDP (POMDP).
A possible solution to this problem is to approximate this POMDP with a K-th order MDP by concatenating the previous states and actions into an augmented state

$$\mathbf{s}_t^{\text{aug}} = [\mathbf{s}_t, \mathbf{s}_{t-1}, \ldots, \mathbf{s}_{t-k}, \mathbf{a}_t, \mathbf{a}_{t-1}, \ldots, \mathbf{a}_{t-k}]. \tag{3.11}$$

an idea closely related to the NARX concept [61].

State Elements based on Analytical Models

The components of the state can be obtained by the rigid body dynamics Equation 3.1 and the analytical model Equations 3.6 and (3.5). Hence, $\mathbf{q}$, $\dot{\mathbf{q}}$ as well as $\dot{\mathbf{l}}$, $\mathbf{l}$, $\dot{\mathbf{p}}$ and $\mathbf{p}$ should be part of the state. Here $\mathbf{l}$ denotes the vector of muscle lengths and $\mathbf{p}$ the vector of current PAM air pressures. Incorporating the pressures $\mathbf{p}$ into the state also helps to resolve hysteresis effects because rising and falling curves can be discriminated according to $p_t \leq p_t^{\text{des}}$. The order of the system determines how many time-derivatives need to be added to the state. While it is clear from rigid body dynamics Equation 3.1 that the robot arm is a second order system, it is not clear what the order of the PAMs as analytical models for PAMs do are not as well established as the rigid body dynamics equations for traditional robots. For instance, [57] indicates that the activation from Equation 3.6 should possibly be modeled as a second order system but don't take the higher order into account to reduce computational load. In addition, for slower movements higher order derivatives might not play such an essential role in the model prediction and can be held out from the state to avoid higher input dimensionality of the GP. For this reason, we test slow and fast trajectory data sets in the experiments Section 2.4 and check how performance is influence with the orders of the systems.

Estimated State Elements

Another issue with time derivatives is that they are not sensed directly but have to be estimated. A common way is to use finite differences $\dot{m}_t = (m_{t+1} - m_{t-1})/2\Delta t$ where Δt is the sampling period and m_t a measurement. However, the noise on $\dot{m}_t$ would in this case multiply. A GP only assumes output noise and would experience major complications. Unfortunately, from our experience, the delay introduced by filtering the sensor values online on the real robot, even with a second order butterworth lowpass filter, substantially impairs control performance. We, hence, test how adding previous joint angles $\mathbf{q}_{t:t-h}$ and pressures $\mathbf{p}_{t:t-o}$ instead of their respective time-derivatives alter the prediction performance without filtering the input training data.

A similar transformation as finite difference is required to attain the lengths of the cables. The lengths of the muscles are $l_{a,b} = l_0 \pm (q/2\pi)r$ where l_0 is the muscle length for $q = 0 \text{ deg}$ assuming a radial joint with fixed radius r and always tensioned cables. Without a question, the information about the lengths of the PAMs are encoded into the joint angle. An interesting question is whether the GP is able to recover such transformations in addition to unobserved dependencies like elongation and slack of the cables. Based on the previous discussion, we propose to test following state compositions

$$\mathbf{s}_t^{\text{noisy}} = [\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{p}_t, \dot{\mathbf{p}}_t], \tag{3.12}$$

$$\mathbf{s}_t^{\text{padded}} = [\mathbf{q}_{t:t-h}, \mathbf{p}_{t:t-o}], \tag{3.13}$$

$$\mathbf{s}_t^{\text{action}} = [\mathbf{s}_t^{\text{x}}, \mathbf{p}_{t:t-h}^{\text{des}}], \tag{3.14}$$

where $\mathbf{s}_t^{\text{x}}$ is a placeholder for either $\mathbf{s}_t^{\text{noisy}}$ or $\mathbf{s}_t^{\text{padded}}$. The state compositions are empirically tested in Section 3.3.

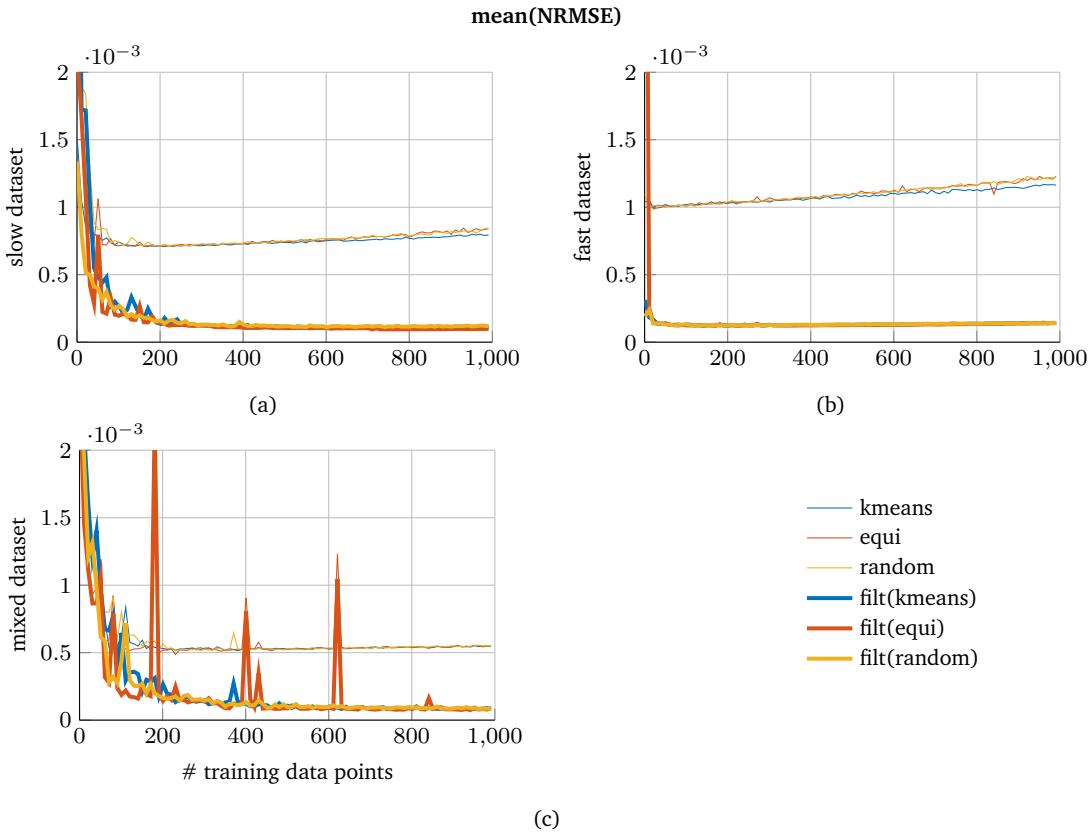


Figure 3.3: The influence of random, k-means and equidistant training data selections model is illustrated with increasing size of training data set on the (a) 'slow', (b) 'fast' and (c) 'mixed' dataset. Experiments have been performed five times and the mean result is depicted. The 'fast' dataset is composed of similar periodic curves and hence needs the least number of data. In case the NRMSE is calculated against unfiltered test targets, the NRMSE tends to rise for more provided data points because the prediction of the GP becomes smoother as the noise modeling improves.

According to [62] a model can only be good enough to fulfill its purpose. [63] coins this aspect purposiveness. The process of disentangling the contribution of the model and the controller to the performance is quite involved. Hence, often long-term predictions [54, 55] are analyzed. This procedure involves a previously collected data sequence consisting of a state $[\mathbf{s}_t]_{t=0}^{T}$ and a corresponding action trajectory $[\mathbf{a}_0]_{t=1}^{T-1}$. Starting with the initial input $\mathbf{x}_0 = [\mathbf{s}_0, \mathbf{a}_0]$ to the GP, the prediction of next state is fed together with the next action as input for the next time step. This iterative procedure is being continued until the predicted state deviates too much from the known state trajectory, e.g. $||\mathbf{s}^{\text{des}} - \mathbf{s}_H||^2 > \epsilon$. The horizon H represents then how well the roll-outs can be simulated and is important for instance for model predictive control type of control frameworks. For muscular systems, the state requires to contain the pressures inside the PAMs. Long-term predictions are then always corrupted by the quality of the pressure model that, in return, is not predefined by the desired trajectory. PAM pressures play a role only when stiffness should be modulated along with the position trajectory, which is not focus of this work. For this reason, we decide to resemble to one-step-ahead predictions and only predict entities that

	slow	fast	mixed
$\mathbf{s}_t^{(1)} = [q_t]$	$2.6e-04 \pm 1.1e-04$	$1.3e-04 \pm 2.2e-06$	$4.8e-04 \pm 2.7e-04$
	$2.6e-04 \pm 3.6e-05$	$1.3e-04 \pm 1.1e-05$	$4.8e-04 \pm 3.5e-04$
	$2.0e-04 \pm 0.0e+00$	$1.3e-04 \pm 0.0e+00$	$3.6e-04 \pm 0.0e+00$
$\mathbf{s}_t^{(2)} = [q_t, q_{t-1}, q_{t-2}]$	$2.8e-04 \pm 3.7e-05$	$1.3e-04 \pm 6.5e-06$	$4.6e-04 \pm 3.0e-04$
	$2.0e-04 \pm 1.3e-05$	$1.3e-04 \pm 3.7e-06$	$3.1e-04 \pm 1.2e-04$
	$2.9e-04 \pm 0.0e+00$	$1.3e-04 \pm 0.0e+00$	$3.1e-04 \pm 0.0e+00$
$\mathbf{s}_t^{(3)} = [q_t, q_{t-1}, q_{t-2}, p_t^a, p_t^b]$	$2.8e-04 \pm 9.0e-05$	$1.3e-04 \pm 2.2e-06$	$3.0e-04 \pm 6.9e-05$
	$2.6e-04 \pm 4.4e-05$	$1.3e-04 \pm 3.4e-06$	$2.8e-04 \pm 8.3e-05$
	$2.9e-04 \pm 0.0e+00$	$1.3e-04 \pm 0.0e+00$	$2.7e-04 \pm 0.0e+00$
$\mathbf{s}_t^{(4)} = [q_t, q_{t-1}, q_{t-2}, p_t^a, p_t^b, p_{t-1}^{a,\text{des}}, p_{t-1}^{b,\text{des}}]$	$2.8e-04 \pm 6.4e-05$	$1.3e-04 \pm 2.6e-06$	$1.7e-04 \pm 9.0e-05$
	$2.3e-04 \pm 3.4e-05$	$1.3e-04 \pm 5.3e-06$	$9.9e-05 \pm 2.7e-05$
	$2.9e-04 \pm 0.0e+00$	$1.3e-04 \pm 0.0e+00$	$2.6e-04 \pm 0.0e+00$
$\mathbf{s}_t^{(5)} = [q_t, \dot{q}_t, p_t^a, p_t^b]$	$3.4e-04 \pm 9.0e-05$	$1.3e-04 \pm 1.3e-05$	$4.3e-04 \pm 1.3e-04$
	$3.0e-04 \pm 3.6e-05$	$1.3e-04 \pm 5.3e-06$	$2.7e-04 \pm 1.0e-04$
	$3.2e-04 \pm 0.0e+00$	$1.3e-04 \pm 0.0e+00$	$3.1e-04 \pm 0.0e+00$

Table 3.1: Normalized root mean squared error from Equation 3.15 calculated for the prediction of each model with different state compositions (rows) against filtered target values from the datasets (columns) depicted in Fig. 3.2. Experiments have been performed five times under same conditions illustrated by the mean NRMSE and one standard deviation. 100 data points have been selected to the training set by random, k-means and equidistant selection. Equidistant selection is deterministic and thus has always zero standard deviation. The best performance on the 'slow' dataset is achieved with $\mathbf{s}_t^{(2)}$ and k-means selection. More complex state representations achieve worse performance. It is clearly visible that for the 'fast' dataset 100 training data points contain enough information and hence is independent of state composition as well as data selection type. This confirms that with enough data the Gaussian Process design choices are less important but evaluation is computationally more expensive. The 'mixed' dataset is harder to model as it is composed of different noise levels in addition to different speeds which require modeling of higher dynamics components. Best performance is achieved with more complex state representations, $\mathbf{s}_t^{(4)}$, as well as k-means selection that is capable to select more distinctive data points compared to the other selection types.

are essential to calculate a control error. Many criterion exist to measure model goodness of one step ahead predictions, see [63] for a detailed description. Here, we employ the normalized root mean squared error

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=2}^{N}(f^{(k)}(\mathbf{s}_{i-1}, \mathbf{a}_{i-1}) - s_i^{(\text{k})})}{N(s_{\text{max}}^{(k)} - s_{\text{min}}^{(k)})}}, \tag{3.15}$$

where $f^{(k)}$ is the k-th element of the prediction and $k \in [1, D]$, $s_{\text{max}}^{(k)}$ and $s_{\text{min}}^{(k)}$ the maximal and minimal value of the state component k in the dataset comprising N data points. The predictions can be test either against the filtered or raw targets from the data set. The filtering of the target signal can be done with a non-causal filter, e.g. zero-phase butterworth lowpass filter which does not add additional delay. Filtered outputs have the benefit to be closer to the latent true data as can be seen in Figure 3.2 (b). We test both cases in Section 3.3.

Filtering Training Outputs

We resort to filter the training outputs although a GP in principle handles output noise. However, we found state dependent noise in our training data set which requires heteroscedastic noise treatment. Figure 3.2 (b) and (d) illustrate that the unfiltered graph changes more randomly for the slow data set than for the fast data set. Heteroscedastic data sets often occur in real applications. Unfortunately, only homoscedastic GPs are analytically tractable rather than a heteroscedastic GP whose inference is based on approximations and training is computationally more demanding. By filtering the training targets, different noise levels of the data set can be balanced to some extend.
Using approaches to handle input noise for GPs [64, 65] is a valid alternative and is subject to future work. In this work, we aim at highlighting problems and give basic answers. Thus, we try to avoid all approximations as quantifying where approximations influence the result is hard to answer.

Subset Selection

Robots generate a lot of correlated data at high frequencies. Keeping all this data in the training data set for non-parametric data is not possible. Hence, the data need to be further subsampled. We do not incorporate approximations to the GP in order to realize bigger training data sets, e.g. with pseudo inputs [58] for reasons stated above.
Hence, we pick a subset of data from the complete data set instead. As a baseline we incorporate random subsampling which is equivalent to guessing. Every approach should improve over random guessing. The default choice is often equidistant subsampling where N data points are picked from the data set that are equally far apart. If, however, the resulting sampling frequency is less than twice the maximum frequency of the time series, some aspects of the signal cannot be captured according to the Nyquist-Shannon sampling theorem. On the other hand, areas where little change happens could be sampled too often. As an alternative, we use k-means algorithm to find N clusters among the complete

data set and pick the data point that are closest to each clusters. Thus, such data points are selected so as to maximize the difference among the training set according to the Euclidean distance.

3.2.3 Gradient-based Control

The control goal is to track a desired trajectory $\{\mathbf{s}_t^{\text{des}}\}_{t=1}^T$ with a control trajectory $\{\mathbf{a}_t\}_{t=1}^T$ with minimal deviation. A simple way to express this desire is to define the squared error

$$e_t^2 = ||(\mathbf{s}_t^{\text{des}} - \mathbf{s}_t)||_Q^2 \,, \tag{3.16}$$

for each time step t. The mean of the forward dynamics model $\bar{\mathbf{f}}$ from Equation 3.7 can be incorporated for the prediction of e_{t+1}^2 in the next time step

$$\begin{aligned} \bar{e}_{t+1}^2 &= (\mathbf{s}_{t+1}^{\text{des}} - \mathbf{s}_{t+1})^T Q (\mathbf{s}_{t+1}^{\text{des}} - \mathbf{s}_{t+1}) \\ \rightarrow \bar{e}_{t+1}^2(\mathbf{a}_t) &= (\mathbf{s}_{t+1}^{\text{des}} - \bar{\mathbf{f}}(\mathbf{s}_t, \mathbf{a}_t))^T Q (\mathbf{s}_{t+1}^{\text{des}} - \bar{\mathbf{f}}(\mathbf{s}_t, \mathbf{a}_t)). \end{aligned} \tag{3.17}$$

The controls in each time step t can than be extracted using

$$\mathbf{a}_t = \underset{\mathbf{a}}{\operatorname{argmin}}\, e_{t+1}^2(\mathbf{a}) \,. \tag{3.18}$$

The main idea of our control framework is to make use of the full posterior distribution that is given by our probabilistic forward model. Thus, we optimize for the expected value of the loss, similar to [66]. In this case Equation 3.16 becomes

$$\begin{aligned} E[e_{t+1}^2(\mathbf{a})] &= {\mathbf{s}_{t+1}^{\text{des}}}^T Q \mathbf{s}_{t+1}^{\text{des}} - 2{\mathbf{s}_{t+1}^{\text{des}}}^T Q E[\mathbf{f}] + E[\mathbf{f}^T Q \mathbf{f}] \\ &= {\mathbf{s}_{t+1}^{\text{des}}}^T Q \mathbf{s}_{t+1}^{\text{des}} - 2{\mathbf{s}_{t+1}^{\text{des}}}^T Q \bar{\mathbf{f}} + \bar{\mathbf{f}}^T Q \bar{\mathbf{f}} + \text{tr}(Q\Sigma) \\ &= \bar{e}_{t+1}^2(\mathbf{a}) + \text{tr}(Q\Sigma(\mathbf{a})) \end{aligned} \tag{3.19}$$

with $[\Sigma]_{i,j} = \text{cov}(f_i, f_j)$ being the covariance of each GP in the forward model vector $\mathbf{f}$ and $\text{diag}(\Sigma) = \sigma^2$. Equation 3.19 can be interpreted as the sum of the mean of the squared error from Equation 3.17 and regularized by the variances of each element of the forward model vector $\mathbf{f}$ weighted by the diagonal elements of Q. This regularization poses a constraint to the overactuation of antagonistic actuation. Hence, by using this control framework, the controls are chosen such that the the successor state is near to the training data set.

3.3 Setup, Experiments and Evaluations

The goal of this chapter is to illustrate how to set up a GP dynamics model and use it subsequently for control of a muscular system. We first introduce the PAM-driven robot arm used to validate our approach. Afterwards, we address the key concepts discussed in section 3.2 with empirical experiments. For the experiments we use the slow, fast and mixed data sets from Figure 3.2 which all have the same size. The slow data set challenges

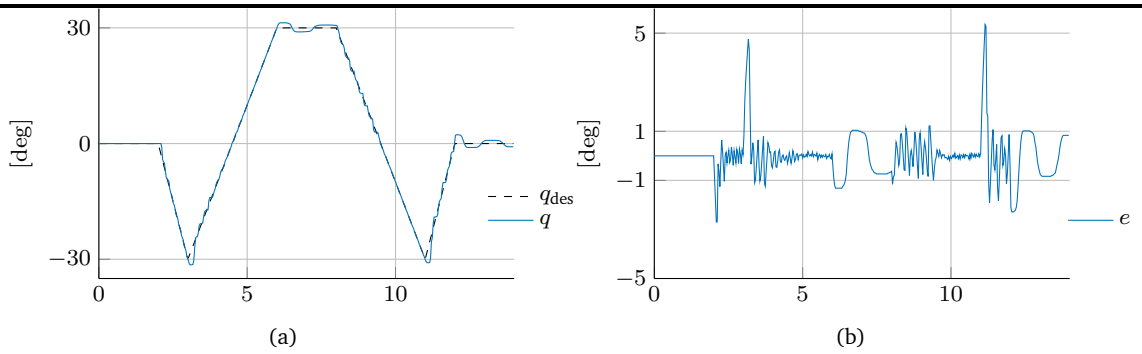


Figure 3.4: Tracking experiment. The controller manages to keep the system in the vicinity of the training data. Also, the controller deals well with hard changes in directions of the desired trajectory and stays apart from that regions within an error bound of ± 1 deg.

the model to accurately distinguish noise from signal. As the relative angle encoders of our muscle based robot have a resolution of $0.036°$, relatively high quantization noise corrupts the signal as can be seen in Figure 3.2 (b) due to quantization. The fast data set consist of rapidly repeating oscillating movement. Thus, more data represents the same curve but higher order dynamics components are excited. The challenge of the mixed data set is its heteroscedastic noise that often occurs in real data sets. We always predict the difference to the next joint angle Δq. Lastly, a tracking task is performed to show that the resulting GP forward model is useful for control.

3.3.1 Experimental Setup: Pneumatic Muscle Robot

We use the PAM-actuated robot arm from [4, 67] which is shown in Figure 2.1. Its key features are a) its lightweight structure with only 700g moving masses, b) powerful PAMs that can lift up to 1.2kN each and generate angular accelerations of up to 28k deg$/s^2$ and c) its design that aims to reduce difficulties for control, e.g., minimal bending of cables etc.. This four DoF robot arm has eight PAMs in total, each two actuating one DoF as an antagonistic pair. Each PAM is equipped with an air pressure sensor and each joint angle is read by an incremental encoder. The pressure in each muscle is changed via Festo proportional valves that regulate the airflow according to the input voltage. All signals are fed into a NI FPGA card that has in turn a C/C++ API for an interface to the main C++ code. Low level PID controller regulate the desired pressures sent from the C++ program within the FPGA. The desired pressures are bounded in the FPGA. In this manner, erroneous signals will not be executed on the real system.

3.3.2 Subset Selection and Model Validation

In this experiment we test how the prediction performance changes with the number of trainings data points as well as subset of data selection strategy. Figure 3.3 shows this experiments for the (a) slow, (b) fast and (c) mixed data sets.
One can observe that in all graphs the mean of the NRMSE decreases when tested against the filtered test targets. When tested against unfiltered test outputs the NRMSE fails at

capturing that the model improves with more data and hence is ruled out as a quality measure of 1-step ahead predictions.
Another observation is that the fast data set can be learned with less training data as less information is contained.
In (c) one can observe that unfortunate equidistant subsampling can lead to bad performance.

3.3.3 Evaluation of State Composition

Figure 3.2 Table 3.1 comprises an experiment where the proposed states and subsampling types from Section 3.2 are tested on the three different data sets from Figure 3.2. The average prediction performance with 100 data points is pointed out together with one standard deviation calculated from five identical experiments for each setting. For the slow data set $\mathbf{s}_t^{(2)}$ with k-means subsampling leads to best performance. Interestingly, more complex states perform worse. The reason is, that for states with higher dimensions more training data is required to fit the model well. Apparently, for simple trajectories a simple state is sufficient.
The fast data set comprises less information compared to the slow data set into more data points. Thus, 100 data points are enough for all states to learn to model well.
The mixed dataset is contains heteroscedastic noise and different curvatures and hence is the most complex compared to the low and fast data set. $\mathbf{s}_t^{(4)}$ with k-means subsampling outperforms by far all other settings. k-means subsampling assures that the most different data points are picked into the training data set. In this manner, the flexibility introduced by state $\mathbf{s}_t^{(4)}$ has a better chance to be utilized.
In general, for rich datasets, a more state should be chosen that enables the model to capture all facets of the underlying data. For simpler datasets the focus can be on the computational load. A smaller state representation can be used in order to speed up computation.

3.3.4 Evaluation of Control Performance

We validate our control approach with a model that 1) filters training outputs, 2) uses k-means clustering to select training data and 3) uses $\mathbf{s}_t^{(4)}$ from table 3.1 as state. The data set is generated by utilizing a PID controller whose performance should be surpassed. In this way we create data that is near to the desired trajectory. The desired trajectory contains edges to show how the controller performs at sudden changes which are especially challenging. Fig. 3.4 shows the tracking experiment. The control performance stays most of the time within a control error bound of ± 1 deg. After the quick changes in reference, the controller manages to return to the desired trajectory quickly.

3.4 Conclusion

Modeling the dynamics of musculoskeletal robots with flexible Gaussian processes is a promising research direction. In this chapter, we identified key problems of using GPs to

replicate the dynamics of muscular systems and proposed appropriate solutions. Subsequently we used this probabilistic model for control.

The contribution of this chapter is twofold: First, we discuss and empirically evaluate how to set up and test a Gaussian process forward dynamics model for muscular systems. We consider different state compositions and subsampling types, debate where filtering makes sense and how to evaluate goodness of our model. Subsequently, we tested our hypotheses on data sets with distinct properties. Second, we show that by using a probabilistic model the overactuation inherent to muscular systems can be resolved while enabling flexible non-parametric models to model the dynamics for such complex systems. We leveraged the uncertainty of the GP to set an additional constraint in the control framework that keeps the system near its training data. The control framework developed in this chapter can be used to tackle the problem of learning fast hitting movements by tracking trajectories optimized through planning algorithms [68, 69].

On the path of illuminating the connection between learning and muscular systems, this work posed an example to the application of Gaussian processes to dynamics modeling of muscular systems. Interesting future directions involve approaches that perform multi-step ahead predictions to improve accurate control further. Including dynamic redundancy resolution, as proposed in this chapter, to multi-step ahead prediction models might enable better models and hence improved control.

Another interesting aspect to extend is to automatically retrieve a Markovian state composition. A possible way would be to use variational autoencoders and develop regularization terms that act on the latent space to This approach is similar to the concept of robotics priors [70].

The classical approach to table tennis – consisting of a planning stage optimizing a target trajectory and a controller tracking this trajectory – includes the assumptions that the generated trajectory is somewhat optimal and that the controller is capable of tracking the trajectory sufficiently well. This are effectively priors on the solution of the task. An intriguing alternative could be to use RL and formulate the table tennis task exclusively in the reward function without taking robot safety into account.

4 Learning to Play Table Tennis From Scratch using Muscular Robots

In the previous chapter, we learned a GP forward dynamics model of the muscular robot and used it to perform a trajectory tracking task. We regularized the control with the uncertainty estimates given by the GP model to resolve the dynamic redundancy while keeping the system close to its previously seen training data. In order to play table tennis using the approach from chapter 2, a trajectory planner is required and the dynamics model would have to predict well in large spaces of states and actions. The prediction time complexity of non-parametric GPs, however, increases linearly with the number of training data points. Hence, tracking fast motions that change substantially remains unsolved. In addition, planning fast motions in the presence of uncertainties arising from difficult ball and robot state estimation is a challenging task.

The task of table tennis can be formulated in a simple manner: the racket needs to hit an incoming ball and the returned ball needs to land on the opponent's side of the table. An alternative approach to the one in Chapter 2, hence, is to define the task purely in a reward function and use Reinforcement Learning to get a policy that performs the table tennis task. Especially, model-free Reinforcement Learning (RL) is a compelling approach to learn complex continuous control tasks. Robot table tennis is an example of a such difficult task. It requires control of fast movements and precise timing. RL could help learning to execute such fast motions from experience. However, to reach a high level of fidelity, model-free RL requires to freely explore and fail in a safe manner for millions of time steps in high velocity areas of the state space. Learning a dynamic tasks directly on real robots, might push the robot to areas of the task space that bring the operational conditions of the system in jeopardy.

The key idea of this chapter is twofold: We propose to leverage the robustness of soft robots to run RL safely on a real robot table tennis task while RL, in turn, learns this complex task directly on real hardware and overcomes the inherent control difficulties of soft robots. In particular, we show that it is possible to learn to return and smash table tennis balls to a desired landing point using a robot arm driven by pneumatic artificial muscles (PAMs) with model-free RL. Our contribution is to illustrate that, using PAM-driven robots, RL can learn table tennis like in simulation, meaning 1) without additional safety considerations 2) while maximizing the speed of the returned balls 3) using a randomly initialized stochastic policy 4) that acts directly on the low level controls of the real system and 5) trains for thousands of trials *from scratch*. We also suggest a practical semi sim and real training to train this task without supervision. To the best of our knowledge, this work pioneers the successful applications of PAM-driven robots to table tennis.

4.1 Introduction

Reinforcement Learning (RL) achieves great performance at ambitious tasks such as the complex game of Go [71], full body locomotion tasks in simulation [72] or difficult continuous control tasks such as robot manipulation [73, 74] to name just a few. All of these hard to solve tasks are either realized in simulation environments or the real robot task is engineered in a way that the RL agent can act like in simulation. In simulated continuous control tasks the agent is allowed to bump into objects, collide with itself and the actions can act on low level controls (such as torques or activations to muscles) while being sampled from a stochastic policy. The policy can be step-based, in contrast to episode-based policies, which permits the agent to react in every time step to changes in state rather than executing an open loop motion. In this manner, the agent freely explores and learns from failures while interacting for millions of time steps with the environment.

Current real world problems to which RL is applied are constraint to merely slow motions such as in manipulation. For such tasks, simple engineered checks assure robot safety. This checks detect for instance 1) collisions with objects (usually based on heuristics) and the robot itself, 2) limit joint accelerations and velocities, 3) stop the robot before it is reaching its joint limits and 4) filter noisy actions generated by stochastic policies [75, 73]. Also, the RL community increasingly recognizes the safety of the robot to be essential if RL should be reliably employed on real robots [76, 77].

Figure 4.1: Our setup consists of a four DoF robot arm actuated by pneumatic muscles and a ballgun that shoots balls towards the robot arm. The robot is supposed to return the ball to a desired landing point on the table. The Cartesian positions of the ball are measured by a color-based camera detection system.

For dynamic tasks like robot table tennis this safety checks are disadvantageous for two reasons: 1) empirically finding parameters for the safety heuristics is substantially harder at faster motions and 2) the safety checks heavily limit the capabilities of robots as they are conservatively chosen in order to reliably avoid damage to the real system. Cautious checks are fine for slow motions as they suffice, e.g. for grasping and lifting an object, but for a task like table tennis it is essential to exert explosive movements. Imagine that an incoming ball is supposed to be hit at a point in space which is far away from the current racket position. In such situation, the agent needs to accelerate the racket rapidly in order to gain momentum and reach the hitting point in time. Thus, the amount of accelerations that the robot is capable of generating is proportional to the upper bound on the dexterity it can reach in table tennis.

To this point, it is challenging to not limit the performance of dynamic real robotic tasks too much by safety check *and* let the agent explore fast motions freely. For this reason,

learning approaches to robot table tennis using anthropomorphic human arm sized systems usually employ techniques such as imitation learning [69], choosing or optimizing from safe demonstrations [78, 68, 37], minimizing acceleration for optimized trajectories [79], distributing torques over all degrees of freedom (DoF) [80] as well as cautious learning control approaches [81].

In [4, 67] it has been shown that systems actuated by pneumatic artificial muscles (PAM) are suitable to execute explosive hitting motions safely. By adjusting the pressure range for each PAM, a fast motion generated by the high forces of the PAMs can be decelerated before a DoF exceeds its allowed joint angle range. In addition, PAM-driven robots are inherently backdrivable which makes them especially robust. However, such actuators are substantial harder to control than traditional motor-driven systems [43, 11]. For this reason, PAM-driven systems are predominantly used as a testbed for control approaches.

In this chapter we show that soft actuation enables RL to be safely applicable to a dynamic task directly on real hardware and, vice versa, RL helps to overcome the difficulties of a dynamic task as well as the control issues of soft robots. In particular, we show that, by using the robot actuated by pneumatic artificial muscles (PAM) from [4, 67], we can learn to return and even smash table tennis balls using RL *from scratch*. Rather than avoiding fast motions, we favor highly accelerated strikes by maximizing the velocity of the returned ball where robot safety is handled inherently by the hardware. In addition, we 1) apply noisy actions sampled from a Gaussian multi layer perceptron (MLP) policy directly on the low level controls (desired pressures), 2) while running the RL algorithm for millions of time steps, 3) randomly initialize the policy at start and 4) introduce a hybrid sim and real training procedure to circumvent practical issues of long duration training such as collecting, removing and supplying table tennis balls from and to the system. Using this training procedure, we learn to return and smash balls without using real balls during training. By enabling RL to act on the real system in the same way as in simulation, we enable to leverage the benefits of such complex systems, despite the control issues, to be the first to learn this challenging task of playing table tennis with muscular robots.

In Section 4.2 we introduce the reward functions that have been incorporated to learn to return and smash as well as the hybrid sim and real training procedure. The return and smash experiment are described in detail in Section 4.3. We summarize the contributions and discuss this chapter in Section 4.4.

4.2 Training of Muscular Robot Table Tennis

The symbiosis of learning dynamic tasks and muscular robots allows the RL algorithm to act in the same way as in simulation while RL helps to overcome the inherent control difficulties PAM-driven systems and leverage its beneficial properties for the task. In this section, we introduce the return and smash table tennis task and illustrate the details of this symbiosis. We further introduce dense reward functions that formulate the intention to return an incoming ball to a target location on the table and to maximize its velocity at impact.

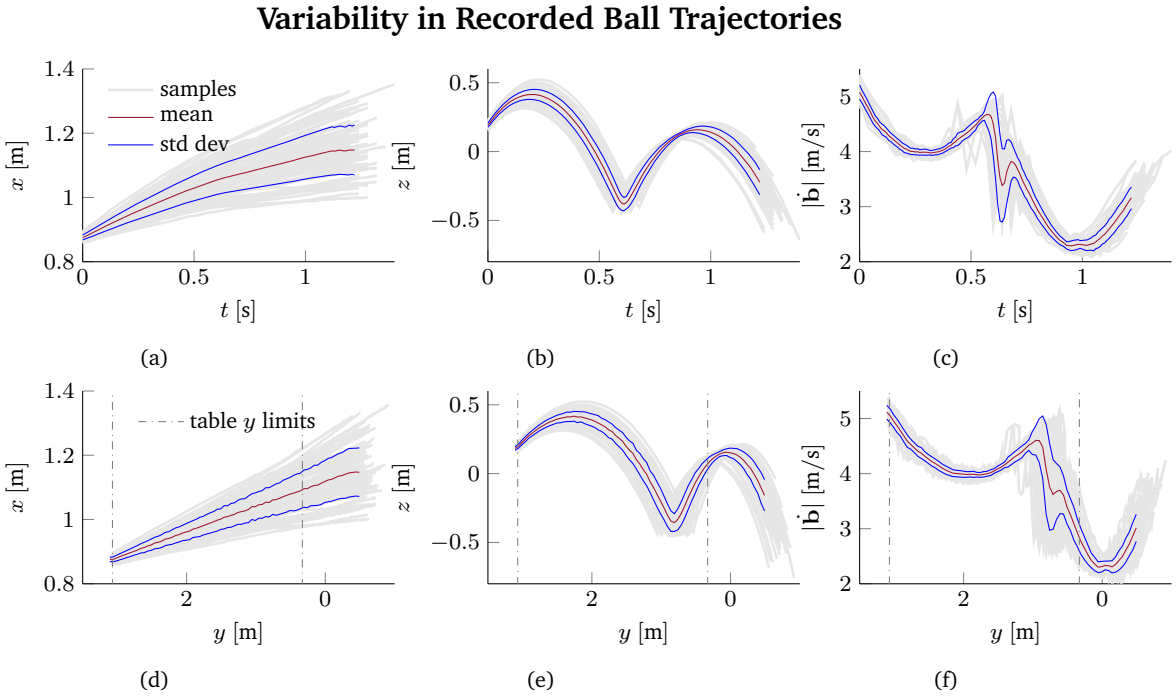


Figure 4.2: Variability of the recorded ball trajectory dataset $\mathcal{D}^{\text{rec}}$. Table tennis balls have been launched by a ball launcher with fixed settings towards the robot. Recording was done via a color-based camera vision system [82]. Variability is quantified by the sample mean and sample variance with respect to time a) to c) and along the long side of the table (y coordinate) d) to f). The shorter edge of the table is aligned with the x coordinate and z is linked to the normal of the table plane. Although the settings of the ball launcher are constant, the ball varies substantially: 1) variability increases over the duration of the trajectory, especially after the bounce, 2) the first bounce on the table (incoming ball) varies around $50\,\text{cm}$ along y-coordinate which can be seen from subfigure e). 3) the agent has to handle deviation of around $40\,\text{cm}$ along the x-axis and $\sim 50\,\text{cm}$ along the z-axis when the ball is in reach for the hit. 4) the variability with respect to time and the y-axis increases over time which means that the agent cannot simply learn to start the hit after a particular fixed duration. Additionally, the agent needs to learn to adjust the amount of energy it transfers to each individual ball as the ball velocity also varies between at the end of the table.

4.2.1 Muscular Robot Table Tennis Task Setup

The table tennis task consists of returning an incoming ball with the racket attached to the robot arm to a desired landing point on the table $\mathbf{b}^{\text{des}} \in \mathbb{R}^2$. We denote the ball trajectory $\boldsymbol{\tau}^{\text{b}} = [\mathbf{s}_t^{\text{b}}]_{t=0}^{T}$ consisting of a series of ball states $\mathbf{s}_t^{\text{b}} = [\mathbf{b}_t, \dot{\mathbf{b}}_t]$ that themselves contain the current ball position $\mathbf{b}_t \in \mathbb{R}^3$ and velocity $\dot{\mathbf{b}}_t \in \mathbb{R}^3$. In a successful stroke, the robot hits the ball at time t_{h} and position $\mathbf{b}^{\text{h}}$ such that the ball lands on the table plane at position $\mathbf{b}^{\text{land}}$ at time t_{land}. The ball crosses the plane aligned with the net on the incoming and outgoing ball trajectory at position $\mathbf{b}^{\text{n}_{\text{in}}} \in \mathbb{R}^2$ at time $t_{\text{n}_{\text{in}}}$ and $\mathbf{b}^{\text{n}_{\text{out}}} \in \mathbb{R}^2$ at time $t_{\text{n}_{\text{out}}}$ if the robot successfully returns the ball.

Table tennis falls into the general class of dynamic tasks such as baseball [83], tennis or hockey [84]. Dynamic tasks represent a class of problems that are relatively easy to solve for humans but hard for robots. The features of dynamic tasks are 1) quick reaction times as some moving object has to be touched or hit in time, 2) precise motions because the object is supposed to arrive in some goal state (e.g. desired landing position on the table) and 3) fast and highly accelerated motions. The latter point is important for two reasons: First, a successful strategy can incorporate a fast strike like in a table tennis smash. Second, highly accelerated motions are required in case the desired hitting position of the ball $\mathbf{b}^{\text{h}}$ is far away from the current racket position. For this reason, the maximum acceleration the robot system is capable of generating represents the upper limit to the dexterity the agent can develop at such tasks. This class of problems can be seen in contrast to robot manipulation where the task itself is richer than a dynamic task in the sense that the objects and setting can vary largely. For this set of tasks, however, slow motions are sufficient.

Particularly useful for dynamic tasks are robots actuated by pneumatic artificial muscles (PAM). This actuators contract if the air pressure inside increases, hence at least two PAMs act antagonistically on one degree of freedom (DOF) as a single PAM can only pull and not push. In this chapter, we leverage the PAM-driven robot arm developed in [4, 67] which has four DoFs actuated by eight PAMs. Such robots are capable of generating high accelerations due to a high power-to-weight ratio. At the same time, allowed pressures ranges can be adjusted such that joint limits are not reached despite fast motions. We use this property in Section 4.3.1 and Section 4.3.2 to let the RL agent freely explore fast motions without further safety considerations. Another benefit of PAM-driven systems is the inherent robustness due to passive compliance. This property helps reducing damage at impact due to shock absorption [85] as well as adjusting stiffness due to co-contraction of PAMs in an antagonistic pair. In this chapter, we leverage the robustness to apply stochastic policies directly on the desired pressures which are the low level actions in this system (see Figure 4.4).

This numerous beneficial properties come at the cost of control difficulties. PAMs 1) are highly non-linear systems that 2) change their dynamics with temperature as well as wear and 3) are prone to hysteresis effects. Thus, modeling such systems for better control is challenging [43, 11]. For this reason, predominately used as testbed for control algorithms rather than utilizing the advantages of this actuators. In this work, we show that it is possible to satisfy the precision demands of the table tennis task despite the control difficulties of PAM-driven systems by using RL (see Section 4.3.1 and Section 4.3.2).

4.2.2 Dense Reward Functions For Returning and Smashing

We formulate the learning problem as an episodic Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \mathcal{P}, p_0, \gamma) \tag{4.1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the immediate reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the transition probability or dynamics, p_0 is the initial state distribution and $\gamma \in [0, 1]$ the discount factor. The goal in RL is to find a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that maximizes the expected return $J = \mathbb{E}_{\boldsymbol{\tau}} \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)$ where $\boldsymbol{\tau} = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T)$ is the state action trajectory, $\mathbf{s}_0 \sim p_0$, $\mathbf{a}_t \sim \pi(\mathbf{s}_t)$ and $\mathbf{s}_{t+1} = \mathcal{P}(\mathbf{s}_t, \mathbf{a}_t)$.
The state $\mathbf{s} = [\mathbf{s}^{\text{b}}, \mathbf{s}^{\text{r}}]$ we use here is composed of the ball state $\mathbf{s}^{\text{b}}$ and the robot state $\mathbf{s}^{\text{r}}$. The robot state $\mathbf{s}^{\text{r}} = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}]$ consists of the joint angles $\mathbf{q} \in \mathbb{R}^4$, joint angle velocities $\dot{\mathbf{q}} \in \mathbb{R}^4$ and air pressures in each PAM $\mathbf{p} \in \mathbb{R}^8$. The ball state $\mathbf{s}^{\text{b}} = [\mathbf{b}, \dot{\mathbf{b}}]$ has been already defined in Section 4.2.1. The system we utilize here actuates each DoF by the minimum number of two PAMs. The actions $\mathbf{a}$ are hence the change in desired pressures in each PAM $\Delta\mathbf{p}^{\text{des}} \in \mathbb{R}^8$. The reader is referred to [67] for more information about the system. In practice, the true Markovian state $\mathbf{s}$ is not accessible in experiments with real robots. Especially for PAM driven systems, the Markov state composition is unclear [11] leading to a Partially Observable MDP (POMDP) which assumes to receive rather observations $\mathbf{o}$ of the true state $\mathbf{s}$. As we are not treating this case in this paper, we continue using $\mathbf{s}$ instead of $\mathbf{o}$ notation for clarity.
The immediate reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ defines goal of the task. The task of returning an incoming ball to a desired landing point can be separated into two stage: 1) manage to hit or touch the ball and 2) fine-tune the impact of the ball with the racket such that the ball flies in a desired manner. As the landing location of the ball is only influenced in case the robot manages to hit the ball, we introduce a conditional reward function

$$r = \begin{cases} r^{\text{tt}} & \text{racket touches the ball} \\ r^{\text{hit}} & \text{otherwise}\,, \end{cases} \tag{4.2}$$

where r^{tt} is the table tennis reward that gives a value to the stroke depending on the ball trajectory after the impact of ball and racket The hitting reward

$$r^{\text{hit}} = -\min_t ||\boldsymbol{\tau}_t^{\text{b}} - \boldsymbol{\tau}_t^{\text{r}}|| \tag{4.3}$$

is a dense reward function representing the minimal Euclidean distance in time between the ball trajectory $\boldsymbol{\tau}^{\text{b}}$ and Cartesian racket trajectory $\boldsymbol{\tau}^{\text{r}}$ where $\boldsymbol{\tau}_t^{\text{r}} = \mathbf{x}_t^{\text{r}} = \mathcal{T}(\mathbf{q}_t) \in \mathbb{R}^3$ using the forward kinematics function $\mathcal{T}(\cdot)$, the Cartesian racket position $\mathbf{x}_t^{\text{r}}$ and ignoring the racket orientation. This reward function encourages the agent to get closer to the ball and finally hit it by providing feedback about how close the ball missed the racket. The table tennis reward

$$r^{\text{tt}} = \begin{cases} 1 - c||\mathbf{b}^{\text{land}} - \mathbf{b}^{\text{des}}||^{\frac{3}{4}} & \text{return task} \\ (1 - c||\mathbf{b}^{\text{land}} - \mathbf{b}^{\text{des}}||^{\frac{3}{4}}) \max_{t > t_{\text{h}}} ||\dot{\mathbf{b}}_t|| & \text{smash task} \end{cases} \tag{4.4}$$

considers the distance of the actual landing point $\mathbf{b}^{\text{land}}$ to the desired landing point $\mathbf{b}^{\text{des}}$ for the return task (see Section 4.3.1). The normalization constant $c = ||\boldsymbol{\tau}_0^{\text{r}} - \mathbf{b}^{\text{des}}||^{-1}$ is chosen such that r^{tt} is usually within the range $[0, 1]$ where $\boldsymbol{\tau}_0^{\text{r}}$ is the initial racket position. We also cap the table tennis reward $r^{\text{tt}} = \max(r^{\text{tt}}, -0.2)$ in order to avoid too negative rewards in case the ball is shot into a random direction with high velocity as happens if hit by an edge of the racket. For the smashing task the agent is supposed to simultaneously maximize the ball velocity $\dot{\mathbf{b}}$. We force the agent to be precise *and* play fast balls by introducing the product between this two goals. In this way, if a single component has a low value, r^{tt} is small overall. We illustrate the efficacy of this reward function in Section 4.3.2. We also introduce an exponent to the components of r^{tt} in order to cause the values of the rewards to be more different closer to the optimal value. We found the exponent $3/4$ to empirically work well.
Note that we do not incorporate *any* safety precautions such as state constraints like joint ranges, minimal accelerations $||\ddot{\mathbf{q}}_t||$ or change in actions $||\mathbf{a}_t - \mathbf{a}_{t-1}||$ into the reward function. On the contrary, we add a term that favors faster hitting motions.

4.2.3 Hybrid Sim and Real Training

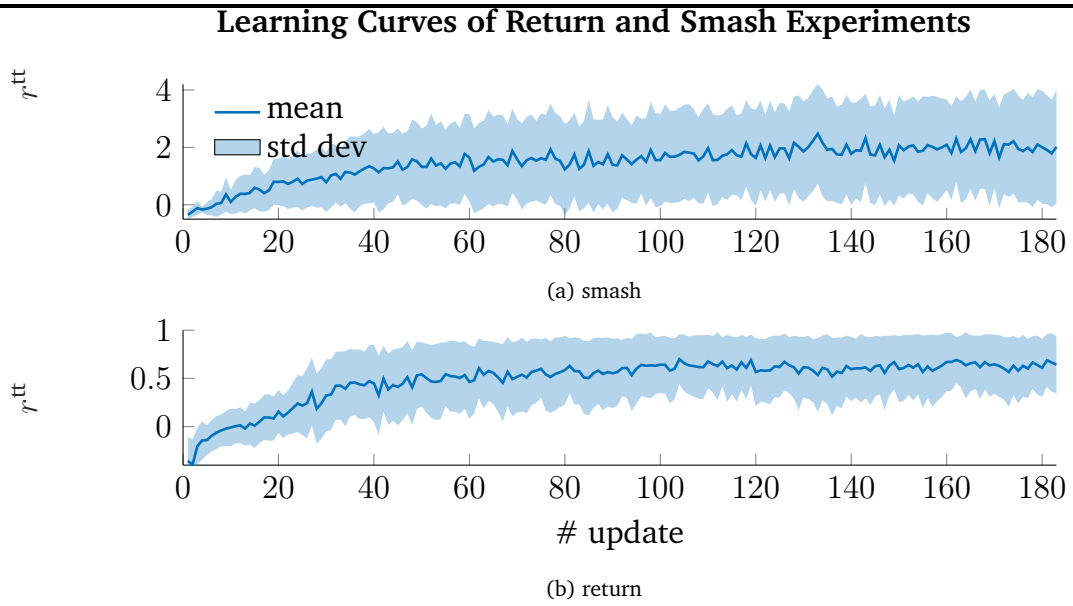
Running experiments with real robots and objects for millions of time steps is a tedious practical effort. In table tennis the ball has to automatically be shot by the ball launcher, removed from the scene after the stroke and be returned to the ball reservoir of the ball launcher. Automating this pipeline takes a substantial amount of work. Hence, we decided to train with simulated balls. Simulated balls, however, might differ from the real ball to an amount that the learned policy is not useful when playing with real balls. For this reason, we record multiple real ball trajectories and replay them in simulation during the training process. Specifically, we collected a data set $\mathcal{D}^{\text{rec}} = [\boldsymbol{\tau}^{\text{rec},i}]_{i=0}^{N^{\text{rec}}}$ using the color-based vision system where the balls were launched by a ball launcher with fixed settings. Within the recorded dataset $\mathcal{D}^{\text{rec}}$ the i-th trajectory consist of a sequence of ball states $\boldsymbol{\tau}_t^{\text{rec},i} = \mathbf{s}_t^{\text{rec}}$. Although the settings on the ball launcher have been kept fixed, the recorded ball trajectories vary largely as can be seen in Figure 4.2. For this reason, we collected $N^{\text{rec}} = 100$ ball trajectories to represent this variability. In every episode we uniformly sample a ball trajectory i that is then replayed.
In case the replayed ball touches the racket during training, the rebound model from [36]

$$\dot{\mathbf{b}}_{\text{out}} - \dot{\mathbf{x}}^{\text{r}}_{t_{\text{h}||}} = \epsilon_{\text{R}}(-\dot{\mathbf{b}}_{\text{in}} + \dot{\mathbf{x}}^{\text{r}}_{t_{\text{h}||}}) \tag{4.5}$$

calculates the outgoing velocity of the ball $\dot{\mathbf{b}}_{\text{out}}$ from the ball velocity $\dot{\mathbf{b}}_{\text{in}}$ before impact, the racket speed $\dot{\mathbf{x}}^{\text{r}}_{t_{\text{h}||}}$ at impact (all measured along the racket normal) and the restitution coefficient of the racket ϵ_{R}. Note that this model assumes no spin. After impact of the ball with the racket, the subsequent ball trajectory is unclear if using the recorded ball dataset $\mathcal{D}^{\text{rec}}$ only. Hence, we continue to simulate the ball given $\dot{\mathbf{b}}_{\text{out}}$ and $\mathbf{b}_{\text{out}}$.
Due to the lack of good models of PAM-driven systems [43, 11], the real robot has been used during training instead of being simulated. The actions $\mathbf{a}$ sampled from the policy π during training have been directly applied onto the real system. The resulting robot state $\mathbf{s}^{\text{r}}$ is read from the sensors and fed back into the simulation and the RL framework. In

Algorithm 2 Hybrid Sim and Real Training Episode

$i \sim \text{uniform}(0, N^{\text{rec}})$
$\boldsymbol{\tau}^{\text{rec}} \leftarrow \mathcal{D}_i^{\text{rec}}$
$t \leftarrow 0$
racketTouched$\leftarrow$*false*
while episode end not reached **do**
  if racket touches ball **then**
    $t_{\text{h}} \leftarrow t$
    $\dot{\mathbf{b}}_{\text{out}} \leftarrow \text{rebound}(\dot{\mathbf{b}}_{\text{out}}, \dot{\mathbf{x}}^{\text{r}}_{t_{\text{h}}||}, \epsilon_{\text{R}})$ (Equation 4.5)
    $\boldsymbol{\tau}^{\text{sim}}_{t>t_{\text{h}}} \leftarrow \text{sim}(t_{\text{h}}, \dot{\mathbf{b}}_{\text{out}}, \mathbf{b}_{\text{out}})$
    racketTouched$\leftarrow$ *true*
  end if
  $\mathbf{s}^{\text{r}}_t \leftarrow$readSensors()
  if racketTouched **then**
    $\mathbf{s}^{\text{sim}}_t \leftarrow \boldsymbol{\tau}^{\text{sim}}_t$
    $\mathbf{s}_t \leftarrow [\mathbf{s}^{\text{r}}_t, \mathbf{s}^{\text{sim}}_t]$
  else
    $\mathbf{s}^{\text{rec}}_t \leftarrow \boldsymbol{\tau}^{\text{rec}}_t$
    $\mathbf{s}_t \leftarrow [\mathbf{s}^{\text{r}}_t, \mathbf{s}^{\text{rec}}_t]$
  end if
  $\mathbf{a}_t \leftarrow \pi(\mathbf{s}_t)$
  $t \leftarrow t + 1$
end while



(a) smash

(b) return

Figure 4.3: Sample mean and standard deviation of the rewards of each episode for the a) smash and b) return task from Section 4.3.1 and Section 4.3.2. The policies for both experiments were updated 183 times. The number of episodes differs for each experiment as we define the end of the training by the number of time steps (1.5 million for both experiments at 10 ms) but the actual episode length varies. For this reason the return task required 15676 and the smash task 15161 strokes/episodes. Each episode takes approximately 1 s with additional initialization of the robot (2 to 4 s, see Figure 4.5) The actual moving time is 14 h and 10 min for the return task and 14 h and 18 min for the smash experiment.

this manner, the real robot moves in simulation the same way as in reality. Algorithm 2 summarizes a single episode of the training procedure. In this manner, the real robot moves in simulation the same way as in reality assuming our forward kinematics model in simulation is accurate.

This practical way of training allows to use the simulation conveniently to estimate the landing position $\mathbf{b}^{\text{land}}$ and the ball velocities $\dot{\mathbf{b}}_{t>t_{\text{h}}}$ that we need for the reward (see Equation 4.3 and Equation 4.4) as well as use the boolean contact indicators provided by MuJoCo. In addition, we avoid collecting and launching balls. Note that this way of training can serve as an entry point for sim2real techniques such as domain randomization. For instance, the ball initial state $\mathbf{s}_0^{\text{b}}$ could have been randomly chosen or the ball trajectory $\boldsymbol{\tau}^{\text{b}}$ perturbed.

4.3 Experiments and Evaluations

The key idea of this chapter is to 1) enable RL to explore fast motions without safety precautions using soft robots and, by doing so, 2) learn a difficult dynamic task with a complicated real system using RL. To show 2), we learn to return and smash a table tennis ball with a PAM-driven robot arm using the practical semi sim and real training procedure from Section 4.2.3 and the reward functions from Section 4.2.2. We highlight 1) by quantifying the robustness of the system during the training. In particular, we illustrate the speed of the returned ball, report maximum accelerations of the racket and depict the noisy actions on the low level controls of the real system due to the application of a stochastic policy. Results are best seen in the supplemental videos at `robotlearning.ai`.

4.3.1 Learning to Return

Returning table tennis balls with PAM-driven systems is a challenging problem due to PAMs being hard to model and control and table tennis requiring precise control at impact of the ball with the racket. We demonstrate that by enabling RL to explore freely at fast paces, the agent is able to learn this task. In particular, the robot is supposed to return balls to a desired landing position $\mathbf{b}^{\text{des}}$ (see Figure 4.7) on the opposite side of the table shot by a ball launcher as can be seen in Figure 4.1 and is described in Section 4.2.1.
We let the robot train for 1.5 million times steps using a stochastic policy. The policy has been randomly initialized and the actions are the change in target pressures. One strike corresponds to one episode and the agent receives a reward according to the dense return reward function from Equation 4.4 at the end of each episode. We use PPO as a backbone RL algorithm. In particular, we leverage the ppo2 implementation of PPO [86] from OpenAI baseline [87]. Table 4.1 lists the other hyperparameters used for this experiment.
After training the final policy has been tested with real balls. The agent managed to hit 96% and return 77% of the 107 real balls that have been shot by the ball launcher as indicated in Table 4.2. Figure 4.7 illustrates how far each landing point was apart from the desired landing point. As we did not directly specify to return to the opponent's side of the table, the landing points are spread in circle around the desired landing point. This circle overlaps with the opponent's side but is not fully contained by it. For this reason, the return rate to the table would be higher if $\mathbf{b}^{\text{des}}$ would have been moved towards the center of the table half.
Interestingly, the agent did not only learn to intercept the ball but also to prepare for the hit as can be seen in Figure 4.5. This two stages are part of the four stages of a table tennis game introduced in [88] and recorded from play in [36]. This behavior emerged although the goal was only to return the ball to a desired landing point. Specifying the same behavior within the classical pipeline of 1) planning a reference trajectory and 2) tracking with an existing (model-based) controller appears to be more problematic. Hence, this work can be seen as a type of end-to-end approach to dynamic tasks where we learn a mapping from sensor information to low level controls directly which is only possible when using the robustness of soft robots.
It is worth mentioning that - by being able to learn this task like in simulation - we enable to put as little priors on the solution as possible. We neither have to add any constraints or regularizers, such as minimal accelerations or energy, nor did we have to use a higher abstraction level like task or joint space but instead learn directly on the low level controls. All regularizers are priors on the solution that the algorithm converges to. By avoiding such priors, the solution emerges purely from the reward function and the hardware. For dynamic tasks such as table tennis on real robots, safety usually has to be taken into account in some form.

4.3.2 Learning to Smash

In table tennis, smashing is a means of maximizing the ball velocity such that the opponent has a hard time returning. The motion needs to be very fast and at the same time precise enough to return the ball to the opponent's side. Being precise when smashing

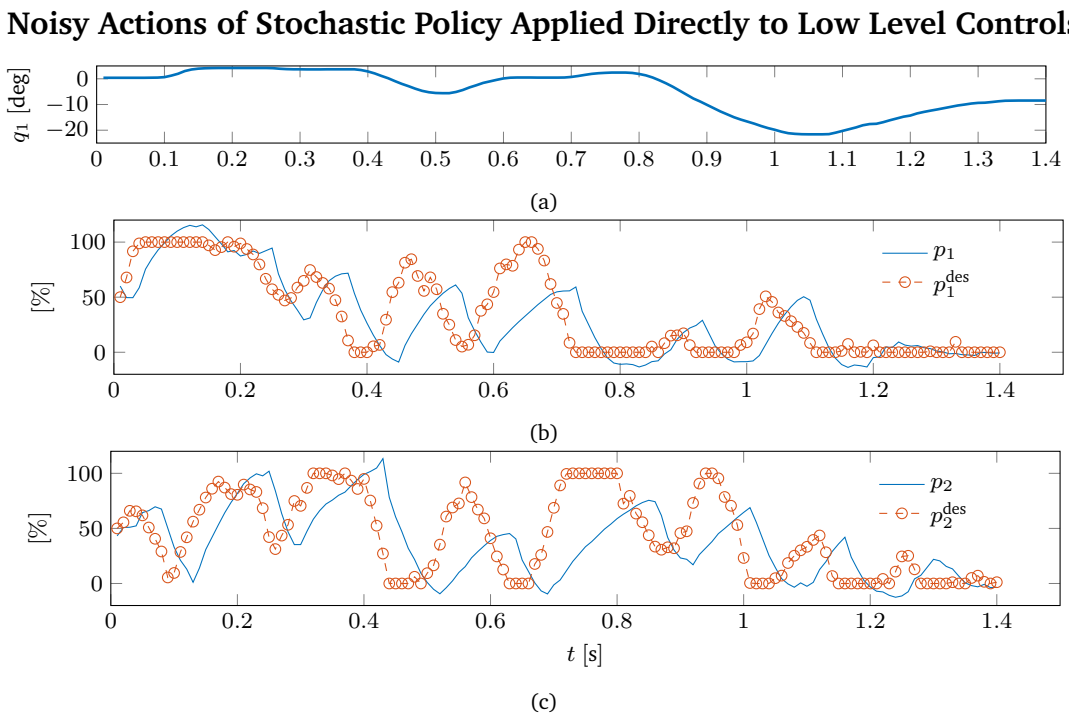


Figure 4.4: Noisy actions due to sampling of a stochastic policy in every time step exemplified on the first DoF with both antagonistic PAMs p_1 and p_2 of one episode. The pressures are normalized between $[0\% \dots 100\%]$ to indicate that pressure ranges which is equivalent to air pressures of $[0\,\text{bar} \dots 3\,\text{bar}]$. The desired pressures p^{des} often switch between almost the whole pressure range multiple times during the episode. The actual pressures follow with some delay. The impact with the ball happens between $t = 0.8\,\text{s}$ and $t = 1\,\text{s}$. Before this interval at $t = 0.7\,\text{s}$ the agent switches pressures from minimum to maximum and vice versa to hit the ball. Applying such action sequences to the low level controls of traditional robots of the same dimensions (e.g. link lengths) as the robot arm we use, likely breaks the system.

(a) Initial posture (b) Preparing for hit (c) Actual hit

Figure 4.5: Images extracted from a video showing the learned hitting motion of the return experiment. After an episode, the robot initializes to a initial position shown in a) which takes $2\,\text{s}$ to $4\,\text{s}$. The agent automatically learned two distinctive hitting stages consisting of b) preparing for a hit and c) the actual hit.

balls is even harder to learn than just returning balls. One reason is that hitting the ball at faster racket paces leads to bigger deviations of the landing point compared to hitting the ball at lower velocities. Hence, small errors in racket orientations might lead to the ball not being returned on the table at all. Realizing this with real robots is difficult as learning is required because of the control difficulties of fast motions and learning requires safe exploration. Exploring such motions on traditional robots dramatically increases the chance of damaging the real system. Here we show that, by using soft robots, we can learn this skill using RL only by defining a reward function that maximizes the ball speed and minimizes the distance to the desired landing point (Equation 4.4). Rather than taking safety into account, we - on the contrary - favor aggressive and explosive motions.

We chose to repeat the experiment from Section 4.3.1 with the same hyperparameters as in Section 4.3.1 but using the reward function with speed bonus from Equation 4.4. The learning curve is depicted in Figure 4.3. In comparison with the learning curve from Section 4.3.1, the smashing task is harder to learn for the agent. The standard deviation of the reward for the smash experiment is clearly higher than in the return task. Also, the precision of the returned balls is lower in the smash experiment as can be seen in Figure 4.7.

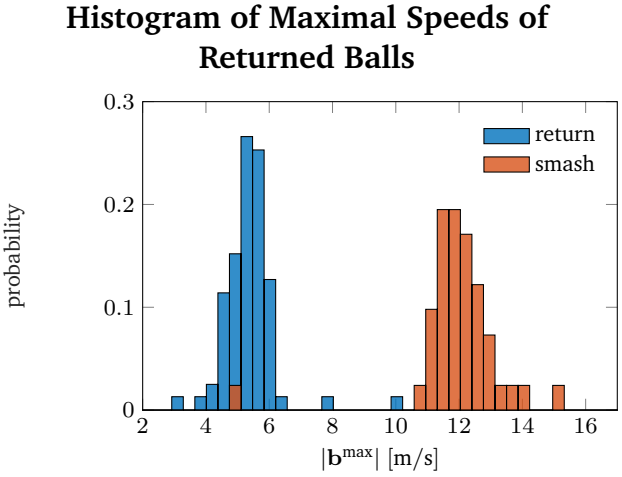


Figure 4.6: Histograms depicting the speeds of the balls for the return and smash experiment. Considered are the maximum velocities after impact with the racket. The final policy of each experiment generated the data for the particular histogram. One can see that the speeds significantly increase for the smashing experiment.

Figure 4.6 shows histograms of the maximum ball velocities after the hit for reward function with and without speed bonus. The ball speed clearly increases when the reward contains a speed bonus. The return and hitting rates indicated in tabtab:rates show that the faster the hit, the less precise the ball can be returned. Hence, the more energy is transferred to the ball, the higher the chance of failing. For this reason, rather mirroring the ball is a substantial easier task than actually smashing it.

4.3.3 Robustness

The robustness of the PAM-driven robot arm enables the RL algorithm to explore in fast motions while executing a stochastic policy directly on the low level controls. We quantify the robustness of this PAM-driven system in multiple manners. First, the sheer number of running a real systems for 1.5 million training time steps stresses that soft robots are indeed robust. 1.5 million time steps at $100\,\mathrm{Hz}$ is equivalent to $14\,\mathrm{h}$ and $10\,\mathrm{min}$ of actual training time. In addition, the robot initializes after each episode which takes further 2 to $4\,\mathrm{s}$ per episode. In total, we train the return task for $14\,\mathrm{h}$ and $18\,\mathrm{min}$ and the smash task for $14\,\mathrm{h}$ and $10\,\mathrm{min}$. Within this durations the policies of both experiments are updated 183 times and we perform 15676 (return)

and 15161 (smash) strokes, each corresponding to one episode. Note that the training has been stopped as the algorithm converged and not due to hardware issues.

Second, each episode is carried out by sampling actions from a Gaussian multilayer perceptron (MLP) in every time-step. For this reason, the signals are noisy and can vary substantially. Figure 4.4 depicts the resulting desired pressures p^{des} from the actions Δp_1^{des} and Δp_2^{des} of the first DoF alongside with the corresponding joint angle q_1 and the measured pressures p_1 and p_2 in percent of the allowed pressure ranges. The hit of the ball happens between $0.8\,\text{s}$ and $1\,\text{s}$. The agent learned to switch the from minimum to maximum pressure and vice versa around $t = 0.7\,\text{s}$ right before the hit. Also, in the preparation phase (see Figure 4.5b) before the hit, the agent used to whole pressure range to bring the robot into a beneficial initial state for the hit. Applying such signals to the low level controls of traditional motor-driven system with the same dimensions as our robot arm presumably causes damage or serious wear.

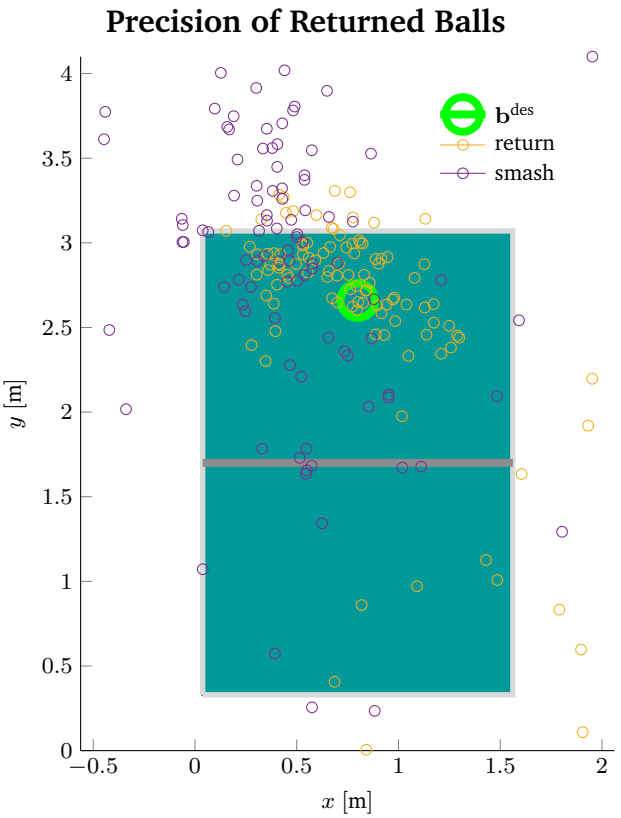


Figure 4.7: The landing points of all returned balls for the return as well as for the smash experiment are shown. The data was generated with the final policies from each experiment. The landing points of the smash experiment are further apart from the desired landing point compared to the return experiment. At the same time this balls are faster, hence, corresponding to a more aggressive but risky game play strategy. A small error in the racket orientation leads to huge deviations in landing point at such speeds. Note that the goal was not to return balls to the tables but return them close to $\mathbf{b}^{\text{des}}$. Hence, moving $\mathbf{b}^{\text{des}}$ more towards the middle point of the opponent's side of the table most likely leads to a higher return rate to the table than reported in Table 4.2.

Third, in both experiment we learn from scratch where the initial policy receives random weights. Still, the motions during training did not exceed the joint limits because the allowed pressure ranges have been set in a way such that one of the muscles in the antagonistic pair is being stretched close to the respective joint limit [4, 67]. In this manner, the robot can train *without* supervision. To achieve safety for dynamic motions on traditional robotic systems, a filter on the actions is required such as in [75]. However, there are multiple downsides to this approach: 1) adding a filter makes state non-Markovian if internal filter state is not part of the RL state which would in turn increases dimensions of state 2) the filter has to be tuned which can be tedious because if the parameters are chosen too conservatively, fast motions are avoided although they are required for the task or, if chosen to optimistically, the robot might be damaged for some configuration, 3) it is counter intuitive to filter the actions that arise from a stochastic policy.

Videos of the whole training of both experiments as well as the evaluation of the final policies we used to calculate statistics in this section are available on `robotlearning.ai`.

4.4 Conclusion

In this chapter, we learn to return and smash table tennis balls with fair amount of precision to a desired landing point using a muscular robot. The synergy between soft robots and Reinforcement Learning for dynamic tasks enabled to achieve this complicated tasks. The robustness of soft actuation allowed RL to act like in simulation and, vice versa, RL helped to overcome the difficulties of the table tennis tasks as well as the control problems of soft robots. In particular, we did not take safety into account although we learned from scratch while even maximizing the speed of the returned ball. Additionally, we introduced a hybrid sim and real training procedure that allows the robot to learn from thousands of strokes without the need to play with real balls.
Although this work is the first in all aspects mentioned above, the performance compared to previous robot table tennis approaches with anthropomorphic robot arms such as in [36] or compared to a human player could not be reached. The most obvious reason is that the robot arm used here is equipped with a total amount of four DoFs whereas the Barrett WAM^{TM} from [36] as well as the human arm have seven DoFs. Another reason is that the training is inefficient as model-free RL is known to be sample-inefficient. We still decided to avoid model-based RL as establishing a model for muscular robots is challenging [11, 43] and learning a policy only is substantially easier. Furthermore, model-free RL algorithms are vulnerable against badly chosen hyperparameters. On the real system, we are limited to manually adapt the hyperparameters whereas in simulation a search can be executed efficiently by running multiple simulations in parallel.
In future work, we aim at learning other table tennis tasks such as serving balls using muscular robots as well as trade-off the speed of the ball with precision of the landing point by introducing a strategy parameter. Overall the goal is to improve the precision of returned balls by overcoming the problems mentioned above.

Table 4.1: Hyperparameters used for RL experiments

hyperparameter	value
algorithm	ppo2
network	mlp
num_layers	1
num_hidden	512
activation	tanh
nsteps	4096
ent_coef	0.001
learning_rate	lambda f:1e-3*f
vf_coefs	0.66023
max_grad_norm	0.05
gamma	0.9999
lam	0.98438
nminibatches	8
noptepochs	32
cliprange	0.4

Table 4.2: Return and hitting rates of return (107 trails) and smash (128 trails) experiments

task	hitting rate	rate of returning to opponents side
return	0.96	0.75
smash	0.77	0.29

5 Conclusion and Future Work

5.1 Summary of Contributions

The research described in this thesis aims to build and leverage soft muscular actuation to extend the capabilities of current robotics approaches to dynamic tasks. The synergy between robot learning approaches and pneumatic muscles has been highlighted in detail throughout this thesis. Soft robots are known for their inherent control and modeling difficulties. Learning can overcome some of these problems. In turn, learning requires - especially for dynamic tasks such as table tennis - to explore the state space to learn from trial and error without breaking the system. Robots actuated by PAMs offer this inherent robustness.

In Chapter 2, we designed a PAM-driven system that is complicated enough to perform interesting tasks (four DoFs actuated by eight PAMs), while avoiding replicating the human anatomy to be as easy to control as possible. In particular, we identified key issues in existing PAM-driven systems, such as PAMs touching each other, the links or PAMs bending over structures, to name just a few. Building upon decades of knowledge of how such robots have been built while not being used for more complex tasks such as table tennis, we show that a simple PID controller is sufficient to track slow trajectories. Moreover, we illustrated the robustness of this system by tuning the parameters of a PID with additional feedforward terms as well as the co-contraction with multi-objective Bayesian optimization (BO). The system can resist the strong forces due to unstable controllers and use it to learn from this bad example rather than prevent such motions. We learned to track a fast trajectory using the tuned controller and found that the optimization converged to similar co-contraction values close to the Pareto front.

Building upon the good control results from Chapter 1, we decided to learn dynamics models from data and use them for control. In particular, we incorporated Gaussian processes (GPs) to learn forward dynamics models as these models assume smoothness of the underlying function that is to be modeled. Thus, GPs are especially interesting for modeling dynamics as dynamical systems are continuous and do not jump. PAM-driven robots still are complicated systems; hence, we identified key issues that can make model learning hard, such as unclear state composition, actuator delay, and subset selection of training data points. The latter point is crucial as using GPs online requires to save only a limited amount of training data points. We show that choosing maximally different data points helps to learn better models. Having a probabilistic GP forward dynamics model, we used it within the Variance-Regularized Control (VRC) framework that we proposed in this chapter. VRC optimizes for the actions of the next time step by incorporating the expected value of the predicted error in the next time step rather than just utilizing the mean prediction. From the math, we find that this is equivalent to using the mean of the error with an additional regularizer that uses the model's variance. This regularization term

pushes the actions to be close to the training data points. This is particularly important for antagonistic systems as the set of pressure combination within an antagonistic muscle pair, that leads to the same joint angle, is infinite. We show that VRC outperforms a hand-tuned PID controller that we used to collect data.
In the previous two chapters, the focus was on perfecting tracking predefined trajectories with the PAM robot built in Chapter 1. However, the ultimate goal is to use the explosive forces PAMs are capable of generating for table tennis. Optimizing useful racket trajectories for table tennis and tracking them with soft robots is a difficult task. In Chapter 3, we rather focus solely on task achievement while leveraging the muscular robots' beneficial properties for dynamic tasks. In particular, we learn to return table tennis balls by defining a reward function that captures only this goal, and we learn this task directly on the real hardware using model-free RL. This procedure is only possible because our system is robust enough to 1) learn for millions of time steps, 2) directly on the low level controls 3) while exploring fast-hitting motions. As we train for $14\,\mathrm{h}$, we propose a hybrid sim and real training procedure where we randomly replay recorded ball trajectories while moving the real system in reality and feeding the sensor information back to the simulation. In this manner, we do not need to train with balls. We show that the robot returns real table tennis balls to a desired landing position on the opponent's side of the table without ever touching a real ball before. Moreover, we show that by maximizing the velocity of the returned ball, the robot learns to smash the ball with $12\,\mathrm{m\,s^{-1}}$ on average ($5\,\mathrm{m\,s^{-1}}$ on average for the return task). This PhD thesis closes nicely by illustrating the harmonic synergy between soft robots and learning approaches discussed at the beginning.

5.2 Discussion and Future Work

During this thesis, we have developed control and modeling approaches as well as hardware for soft robots. In this section, we critically review open problems and promising directions for future research.

Sim2real for Soft Robots

Leveraging simulation dramatically reduces the amount of real robot time. Although soft robots are inherently robust to train for multiple hours, pretraining in simulation would allow achieving better performance given the same amount of real robot time. Sim2real approaches work mostly on traditional motor-driven systems. Developing such approaches for complex soft robots is a big challenge with high stakes if it can be achieved.

Quantifying Dynamic Compliance

Quantifying stiffness or its inverse compliance, is hard if force models are not accurate enough. Learning to evaluate and adapt compliance during motion for the system developed in this thesis would enable us to use this property to adapt it to a given task. It is intriguing to ask if varying stiffness for a table tennis task might help reach higher performance.

In case the compliance can be adapted precisely even during fast motions, it can be used to test the hypothesis that higher stiffness levels lead to less variability in the motion. This hypothesis has already been proven for human arm motions. Doing so for robotic systems opens up to trade-off energy and precision during a hitting motion. In particular, allowing more variability if the ball is far away from the racket might lead to even more precise hitting motions when increasing the stiffness at hitting time.

5.3 Outlook

In this work, we illustrated numerous examples of how muscular robots can be beneficial for dynamic tasks. We hope to convey that good performance in a robotics task is not achieved *despite* but *because* a muscular system has been used.

Bibliography

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *arXiv:1712.01815 [cs]*, Dec. 2017. arXiv: 1712.01815.

[3] S. Kawakami, M. Ikumo, and T. Oya, "Omron table tennis robot forpheus," tech. rep., 2016.

[4] D. Büchler, H. Ott, and J. Peters, "A Lightweight Robotic Arm with Pneumatic Muscles for Robot Learning," in *International Conference on Robotics and Automation (ICRA)*, (Stockholm), May 2016.

[5] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, "Learning To Play Table Tennis from Scratch using Muscular Robots," *IEEE Transaction on Robotics*, 2019.

[6] J. Kober and J. Peters, "Reinforcement Learning in Robotics: A Survey," in *Learning Motor Skills*, vol. 97, pp. 9–67, Cham: Springer International Publishing, 2014.

[7] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*, pp. 908–919, 2017.

[8] M. N. Zeilinger, M. Morari, and C. N. Jones, "Soft constrained model predictive control with robust stability guarantees," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1190–1202, 2014.

[9] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, Mass.: MIT Press, 2006. doi:10.7551/mitpress/3206.001.0001.

[10] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 465–472, 2011.

[11] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters, "Control of Musculoskeletal Systems using Learned Dynamics Models," *IEEE Robotics and Automation Letters*, 2018.

[12] E. P. Zehr and D. G. Sale, "Ballistic movement: muscle activation and neuromuscular adaptation," *Canadian Journal of applied physiology*, vol. 19, no. 4, pp. 363–378, 1994.

[13] D. Driess, H. Zimmermann, S. Wolfen, D. Suissa, D. Haeufle, D. Hennes, M. Toussaint, and S. Schmitt, "Learning to Control Redundant Musculoskeletal Systems with Neural Networks and SQP: Exploiting Muscle Properties," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6461–6468, May 2018.

[14] G. K. H. S. L. Das, B. Tondu, F. Forget, J. Manhes, O. Stasse, and P. Souères, "Controlling a multi-joint arm actuated by pneumatic muscles with quasi-DDP optimal control," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 521–528, IEEE, 2016.

[15] A. Rezoug, B. Tondu, and M. Hamerlain, "Experimental Study of Nonsingular Terminal Sliding Mode Controller for Robot Arm Actuated by Pneumatic Artificial Muscles," 2014.

[16] C. Hartmann, J. Boedecker, O. Obst, S. Ikemoto, and M. Asada, "Real-Time Inverse Dynamics Learning for Musculoskeletal Robots based on Echo State Gaussian Process Regression.," in *Robotics: Science and Systems*, 2012.

[17] S. Ikemoto, Y. Nishigori, and K. Hosoda, "Direct teaching method for musculoskeletal robots driven by pneumatic artificial muscles," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3185–3191, May 2012.

[18] K. K. Ahn and H. P. H. Anh, "Design and implementation of an adaptive recurrent neural networks (ARNN) controller of the pneumatic artificial muscle (PAM) manipulator," *Mechatronics*, vol. 19, pp. 816–828, Sept. 2009.

[19] D. Shin, I. Sardellitti, Y.-L. Park, O. Khatib, and M. Cutkosky, "Design and Control of a Bio-inspired Human-friendly Robot," *The International Journal of Robotics Research*, Nov. 2009.

[20] M. V. Van Damme, B. Vanderborght, B. Verrelst, R. V. Ham, F. Daerden, and D. Lefeber, "Proxy-based Sliding Mode Control of a Planar Pneumatic Manipulator," *The International Journal of Robotics Research*, vol. 28, pp. 266–284, Feb. 2009.

[21] Festo, "Airic's arm," 2007.

[22] T. D. C. Thanh and K. K. Ahn, "Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network," *Mechatronics*, vol. 16, pp. 577–587, Nov. 2006.

[23] A. Hildebrandt, O. Sawodny, R. Neumann, and A. Hartmann, "Cascaded control concept of a robot with two degrees of freedom driven by four artificial pneumatic muscle actuators," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 680–685 vol. 1, 2005.

[24] B. Tondu, S. Ippolito, J. Guiochet, and A. Daidie, "A Seven-degrees-of-freedom Robot-arm Driven by Pneumatic Artificial Muscles for Humanoid Robots," *The International Journal of Robotics Research*, vol. 24, pp. 257–274, Apr. 2005.

[25] I. Boblan, R. Bannasch, H. Schwenk, F. Prietzel, L. Miertsch, and A. Schulz, "A Human-Like Robot Hand and Arm with Fluidic Muscles: Biologically Inspired Construction and Functionality," in *Embodied Artificial Intelligence* (F. Iida, R. Pfeifer, L. Steels, and Y. Kuniyoshi, eds.), no. 3139 in Lecture Notes in Computer Science, pp. 160–179, Springer Berlin Heidelberg, Jan. 2004. doi: 10.1007/978-3-540-27833-7_12.

[26] B. Tondu and P. Lopez, "Modeling and control of McKibben artificial muscle robot actuators," *IEEE Control Systems*, vol. 20, no. 2, pp. 15–38, 2000.

[27] D. Caldwell, N. Tsagarakis, D. Badihi, and G. Medrano-Cerda, "Pneumatic muscle actuator technology: a light weight power system for a humanoid robot," in *1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings*, vol. 4, pp. 3053–3058 vol.4, 1998.

[28] D. G. Caldwell, G. A. Medrano-Cerda, and M. Goodwin, "Control of pneumatic muscle actuators," *Control Systems, IEEE*, vol. 15, no. 1, pp. 40–48, 1995.

[29] C.-P. Chou and B. Hannaford, "Measurement and modeling of McKibben pneumatic artificial muscles," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 90–102, 1996.

[30] G. Klute, J. Czerniecki, and B. Hannaford, "McKibben artificial muscles: pneumatic actuators with biomechanical intelligence," in *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 1999. Proceedings*, pp. 221–226, 1999.

[31] B. Tondu and S. Diaz, "McKibben artificial muscle can be in accordance with the Hill skeletal muscle model," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006*, pp. 714–720, 2006.

[32] C.-P. Chou and B. Hannaford, "Static and dynamic characteristics of McKibben pneumatic artificial muscles," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 281–286, IEEE, 1994.

[33] J. Zhong, J. Fan, Y. Zhu, J. Zhao, and W. Zhai, "One Nonlinear PID Control to Improve the Control Performance of a Manipulator Actuated by a Pneumatic Muscle Actuator," *Advances in Mechanical Engineering*, vol. 6, p. 172782, May 2014.

[34] B. Tondu, "Robust and Accurate Closed-Loop Control of McKibben Artificial Muscle Contraction with a Linear Single Integral Action," *Actuators*, vol. 3, pp. 142–161, June 2014.

[35] G. Andrikopoulos, G. Nikolakopoulos, I. Arvanitakis, and S. Manesis, "Piecewise Affine Modeling and Constrained Optimal Control for a Pneumatic Artificial Muscle," *IEEE Transactions on Industrial Electronics*, vol. 61, pp. 904–916, Feb. 2014.

[36] K. Mülling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behavior*, vol. 19, pp. 359–376, Oct. 2011.

[37] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 650–655, Nov. 2016.

[38] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR Tuning Based on Gaussian Process Optimization: Early Experimental Results," 2015.

[39] S. Bansal, R. Calandra, T. Xiao, S. Levine, and C. J. Tomlin, "Goal-Driven Dynamics Learning via Bayesian Optimization," Mar. 2017.

[40] R. Antonova, A. Rai, and C. G. Atkeson, "Sample efficient optimization for learning controllers for bipedal locomotion," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2016.

[41] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics," Feb. 2016.

[42] Igus, "Robolink light-weight robotic arm," 2015.

[43] B. Tondu, "Modelling of the McKibben artificial muscle: A review," *Journal of Intelligent Material Systems and Structures*, vol. 23, pp. 225–253, Feb. 2012.

[44] E. Kelasidi, G. Andrikopoulos, G. Nikolakopoulos, and S. Manesis, "A survey on pneumatic muscle actuators modeling," in *2011 IEEE International Symposium on Industrial Electronics (ISIE)*, pp. 1263–1269, June 2011.

[45] C. Brosilow and B. Joseph, *Techniques of model-based control*. Prentice Hall Professional, 2002.

[46] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*. Springer Science & Business Media, 1989. doi:10.1007/978-94-009-0909-0_1.

[47] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, vol. 21, pp. 345–383, Dec. 2001.

[48] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. d. Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, pp. 148–175, Jan. 2016.

[49] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1-2, pp. 5–23, 2016.

[50] K. Hosoda, S. Sekimoto, Y. Nishigori, S. Takamuku, and S. Ikemoto, "Anthropomorphic Muscular–Skeletal Robotic Upper Limb for Understanding Embodied Intelligence," *Advanced Robotics*, vol. 26, pp. 729–744, Jan. 2012.

[51] P. Polygerinos, N. Correll, S. A. Morin, B. Mosadegh, C. D. Onal, K. Petersen, M. Cianchetti, M. T. Tolley, and R. F. Shepherd, "Soft Robotics: Review of Fluid-Driven Intrinsically Soft Devices; Manufacturing, Sensing, Control, and Applications in Human-Robot Interaction," *Advanced Engineering Materials*, vol. 19, Dec. 2017.

[52] R. Blickhan, A. Seyfarth, H. Geyer, S. Grimmer, H. Wagner, and M. Günther, "Intelligence by mechanics," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 199–220, 2007.

[53] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[54] A. Doerr, C. Daniel, D. Nguyen-Tuong, A. Marco, S. Schaal, T. Marc, and S. Trimpe, "Optimizing Long-term Predictions for Model-based Policy Search," in *Conference on Robot Learning*, pp. 227–238, 2017.

[55] S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman, "Identification of Gaussian process state space models," in *Advances in Neural Information Processing Systems*, pp. 5315–5325, 2017.

[56] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters, "Stability of Controllers for Gaussian Process Dynamics," *Journal of Machine Learning Research*, vol. 18, no. 100, pp. 1–37, 2017.

[57] F. E. Zajac, "Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control.," *Critical reviews in biomedical engineering*, vol. 17, pp. 359–411, Dec. 1988.

[58] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*, pp. 1257–1264, MIT press, 2006.

[59] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[60] Richard S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[61] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[62] L. Ljung, *System identification: theory for the user*. Prentice Hall PTR, USA, 1999.

[63] J. Kocijan, "System Identification with GP Models," in *Modelling and Control of Dynamic Systems Using Gaussian Process Models*, Advances in Industrial Control, pp. 21–102, Springer International Publishing, 2016.

[64] A. McHutchon and C. E. Rasmussen, "Gaussian process training with input noise," in *Advances in Neural Information Processing Systems*, pp. 1341–1349, 2011.

[65] A. C. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational inference for latent variables and uncertain inputs in Gaussian processes," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1425–1486, 2016.

[66] D. Sbarbaro, R. Murray-Smith, and A. Valdes, "Multivariable Generalized Minimum Variance Control Based on Artificial Neural Networks and Gaussian Process Models," in *Advances in Neural Networks - ISNN 2004*, Lecture Notes in Computer Science, pp. 52–58, Springer, Berlin, Heidelberg, Aug. 2004.

[67] D. Büchler, R. Calandra, and J. Peters, "Learning to Control Highly Accelerated Ballistic Movements on Muscular Robots," *arXiv:1904.03665 [cs]*, Apr. 2019. arXiv: 1904.03665.

[68] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, pp. 263–279, Mar. 2013.

[69] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and Robust Learning of Probabilistic Movement Primitives," *arXiv:1808.10648 [cs, stat]*, Aug. 2018. arXiv: 1808.10648.

[70] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, pp. 407–428, Oct. 2015.

[71] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–359, Oct. 2017.

[72] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of Locomotion Behaviours in Rich Environments," *arXiv:1707.02286 [cs]*, July 2017. arXiv: 1707.02286.

[73] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates," *arXiv:1610.00633 [cs]*, Oct. 2016. arXiv: 1610.00633.

[74] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning Dexterous In-Hand Manipulation," *arXiv:1808.00177 [cs, stat]*, Aug. 2018. arXiv: 1808.00177.

[75] D. Schwab, T. Springenberg, M. F. Martins, T. Lampe, M. Neunert, A. Abdolmaleki, T. Hertweck, R. Hafner, F. Nori, and M. Riedmiller, "Simultaneously Learning Vision and Feature-based Control Policies for Real-world Ball-in-a-Cup," *arXiv:1902.04706 [cs, stat]*, Feb. 2019. arXiv: 1902.04706.

[76] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile QT-Opt for Risk-Aware Vision-Based Robotic Grasping," *arXiv:1910.02787 [cs, stat]*, Oct. 2019. arXiv: 1910.02787.

[77] A. Majumdar, S. Singh, A. Mandlekar, and M. Pavone, "Risk-sensitive Inverse Reinforcement Learning via Coherent Risk Models," in *Robotics: Science and Systems*, 2017.

[78] K. Muelling, J. Kober, and J. Peters, "Learning table tennis with a Mixture of Motor Primitives," in *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 411–416, Dec. 2010.

[79] O. Koç, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robotics and Autonomous Systems*, vol. 105, pp. 121–137, July 2018.

[80] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *2010 IEEE International Conference on Robotics and Automation*, pp. 853–858, May 2010.

[81] O. Koç, G. Maeda, and J. Peters, "Optimizing the Execution of Dynamic Robot Movements With Learning Control," *IEEE Transactions on Robotics*, vol. 35, pp. 909–924, Aug. 2019.

[82] S. Gomez-Gonzalez, Y. Nemmour, B. Schölkopf, and J. Peters, "Reliable Real Time Ball Tracking for Robot Table Tennis," *arXiv:1908.07332 [cs]*, Aug. 2019. arXiv: 1908.07332.

[83] T. Senoo, A. Namiki, and M. Ishikawa, "Ball control in high-speed batting motion using hybrid trajectory generator," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1762–1767, May 2006.

[84] G. Neumann, C. Daniel, A. Paraschos, A. Kupcsik, and J. Peters, "Learning Modular Policies for Robotics," *Frontiers in Computational Neuroscience*, vol. 8, p. 62, 2014.

[85] K. Narioka, T. Homma, and K. Hosoda, "Humanlike ankle-foot complex for a biped robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 15–20, Nov. 2012.

[86] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, July 2017. arXiv: 1707.06347.

[87] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Openai baselines," *GitHub, GitHub repository*, 2017.

[88] Ramanantsoa and Durey, "Towards a stroke Construction Model," *ITTF Education*, Jan. 1994.

Figures and Tables

List of Figures

List of Tables

Notation

The following tables give an overview of the notation and list most of the symbols and acronyms used throughout the thesis. Symbols that only pertain to a specific section are defined where they are used.

Notation	Description
x	scalar
$\{x^{[i]}\}_{i=1}^{N}$	set of N elements x_1 through x_N
$\mathbf{x} = [x_1, x_2, \ldots, x_N]$	vector of N elements
x_i	i-th element of the vector $\mathbf{x}$
$\|\|\mathbf{x}\|\|$	L_2 norm
$\mathbf{x}_{1:N}$	series of N vectors $\mathbf{x}_1$ through $\mathbf{x}_N$
$\mathbf{X}$	matrix
$\mathbf{X}^\top$	transpose of a matrix
$p(x)$	probability density
$\mathbb{E}_{p(x)} f(x)$	expectation of $f(x)$ over $p(x)$
$\nabla_x f(x)$	partial derivative of $f(x)$ w.r.t. x

Symbols	Description
$\mathcal{L}$	loss function
$\boldsymbol{\theta}$	parameter
$\mathbf{s}$	state
$\mathbf{o}$	observation
$\mathbf{a}$	action
r	reward function
γ	discount factor
π	policy
J	expected return
$\boldsymbol{\tau}$	trajectory
$\mathbf{p}$	air pressures
$\mathbf{p}^{\text{min}}$, $\mathbf{p}^{\text{max}}$	minimum and maximum allowed pressures
$\mathbf{b}$	ball position
$\dot{\mathbf{b}}$	ball velocity

Acronym	Description
PAM	pneumatic artificial muscle
GP	Gaussian process
BO	Bayesian optimization
PF	Pareto front

6 Publication List

Journals

Büchler D., Guist S., Calandra R., Berenz V., Schölkopf B., Peters J., 'Learning to Play Table Tennis From Scratch using Muscular Robots', *submitted to IEEE Transactions on Robotics (T-RO)*, 2019

Büchler D., Calandra R. , Peters J., 'Learning to Control Highly Accelerated Ballistic Movements on Muscular Robots', *submitted to Robotics and Autonomous Systems*, 2019

Büchler D., Calandra R. , Schölkopf B. , Peters J., 'Control of Musculoskeletal Systems using Learned Dynamics', *IEEE Robotics and Automation Letters & IROS*, 2018

Conference paper

Büchler D., Ott H., Peters J., 'A lightweight Robotic Arm with Pneumatic Muscles for Robot Learning', *IEEE International Conference on Robotics and Automation (ICRA)*, 2016

Huang Y., **Büchler D.**, Koc O., Schölkopf B., Peters J., 'Jointly Learning Trajectory Generation and Hitting Point Prediction in Robot Table Tennis', *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016

Workshop paper

Büchler D., Calandra R., Peters J., 'Modeling Variability of Musculoskeletal Systems with Heteroscedastic Gaussian Processes', *Neural Information Processing Systems (NIPS) Workshop on Neurorobotics*, 2016

7 Curriculum Vitae

Dieter Büchler

Email : dieter.buechler@tuebingen.mpg.de

Mobile : +49 174 1775241

Current Position

PhD student at the Max Planck Institute for Intelligent Systems *since Aug. 2014*
I am a PhD student in the Empirical Inference department at the MPI for Intelligent Systems in Tübingen. My research focuses on Robotics, Machine Learning, Learning Control approaches to robots actuated by Pneumatic Artificial Muscles. My dream is to enable robots to reach human-level perfection at tasks that are difficult for humans.

Education

- **Max Planck Inst. for Intelligent Systems, Tübingen (Germany)***Aug. 2014 – present*
  PhD student supervised by Prof. Jan Peters, in the dept. Empirical Inference (Prof. Bernhard Schölkopf). I work in the field of Robot Learning on Robots actuated by pneumatic muscles.
- **Imperial College London (UK)** *Oct. 2012 – Sept. 2013*
  Master of Science in Biomedical Engineering; graduated with Distinction
  Main modules: Machine Learning, Robotics, Computational Neuroscience, Image Processing, Biomedical Imaging
- **University of Applied Science Hamburg (Germany)** *Sept. 2007 – Apr. 2012*
  Bachelor of Engineering in Electrical and Information Engineering; 'very good' (ETCS:'A')
  Competitive dual program that combines the B.Eng. degree with an apprenticeship as electronic technician at Siemens Healthcare
- **Siemens Healthcare Hamburg (Germany)** *Sept. 2007 – June 2010*
  Apprenticeship as Electronic Technician.
  Part of the dual degree at the University of Applied Science Hamburg

Research Experience

- **X, the Moonshot factory, Mountain View (USA)** *2018*
  Four months research internship; formerly Google X
  Force based robot control using Machine Learning techniques

- **Imperial College London (UK)** *2013*
  Master thesis
  'Reinforcement Learning for Artificial Muscle Limbs': Built a robotic arm actuated by pneumatic muscles that mimics the human arm in the 2D-plane. Applied Reinforcement Learning approaches to learn the highly nonlinear control.

- **University of Applied Science Hamburg (Germany)** *2012*
  Bachelor thesis
  'Optimale Trajektorien mit Reinforcement Learning': Chose and implemented Neural Fitted Q-Learning to learn to drive as fast as possible over a racing track given similar features as a human driver would have.

- **Siemens Healthcare, Erlangen (Germany)** *2010*
  Six months research internship
  'Magnetic Guided Capsule Endoscopy': Chose and implemented a segmentation algorithm for images taken by the capsule inside the stomach.

Work Experience

- **Siemens Healthcare, Hamburg (Germany)** *2010 – 2014*
  Costumer Service Engineer for Magnetic Resonance Imaging
  Released from duties for the MSc course at the Imperial College London.

- **University of Applied Science Hamburg (Germany)** *2008 – 2011*
  Tutor for Calculus
  Gave summaries of the class and showed how to solve associated exercises.

Student Supervision

- **Simon Guist** *2018*
  Master thesis: 'Reinforcement Learning for Musculoskeletal Systems'

Review Experience

- **EEE Robotics and Automation Letters (RA-L)**: 2017
- **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**: 2017, 2018, 2019
- **Robotics: Science and Systems (R:SS)**: 2016
- **IEEE-RAS International Conference on Humanoid Robots (Humanoids)**: 2018
- **IEEE International Conference on Automation Science and Engineering (CASE)**: 2018, 2019
- **IEEE International Conference on Soft Robotics (RoboSoft)**: 2019
- **IEEE/SICE International Symposium on System Integration (SII)**: 2020

Honors and Awards

- **Award of the Family Klee foundation** *2014*
  Awarded as support for my PhD at the MPI for Intelligent Systems; s-fk.de
- **Foreign studies scholarship of the German National Academic Foundation** *2012*
  Awarded as support for my studies at Imperial College London
- **Member of the German National Academic Foundation** *2010 – 2013*
  Germany's largest and most prestigious academic organization: nominated and assessed

Skills

- **Languages**: German (mother-tongue), English (fluent), Russian (basic)
- **Programming skills**: C, C++, Python, Matlab(Simulink) (proficient), Tensorflow, Java, VHDL, Labview (prior experience)

References

- **Prof. Dr. Jan Peters**: Main supervisor PhD; mail@jan-peters.net
- **Prof. Dr. Aldo Faisal**: Supervisor master's thesis; a.faisal@imperial.ac.uk
- **Prof. Dr. Andreas Meisel**: Supervisor bachelor's thesis; andreas.meisel@haw-hamburg.de