EPFL

Thèse n° 9747

Advancing the Adaptability of Compliant Robot Controllers for Exploration, Interaction, and Manipulation

Présentée le 2 décembre 2022

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'algorithmes et systèmes d'apprentissage
Programme doctoral en robotique, contrôle et systèmes intelligents

pour l'obtention du grade de Docteur ès Sciences

par

Farshad KHADIVAR

Acceptée sur proposition du jury

Prof. A. M. Alahi, président du jury
Prof. A. Billard, directrice de thèse
Prof. D. Prattichizzo, rapporteur
Dr L. Jamone, rapporteur
Prof. J. Hughes, rapporteuse

École polytechnique fédérale de Lausanne

2022

We are what we repeatedly do.
Excellence, then, is not an act, but a habit.
— Aristotle

To those who gift a sense to my life, my family...

Acknowledgements

As I finish this dissertation, I am closing a remarkable chapter of my life. I feel very blessed for the past four years and for meeting many great people without whom I would not have been able to realize my Ph.D.

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Aude Billard, for allowing me to conduct research in her lab. I immensely appreciate her patience during the early days of my Ph.D. and value her endless support, constructive feedback, and high expectations for my work. Above all, I genuinely respect Aude for imparting her knowledge and coaching me toward being a more potent researcher.

I also acknowledge the jury members, Prof. Domenico Prattichizzo, Dr. Lorenzo Jamone, Prof. Josie Hughes, and Prof. Alexandre Alahi, for participating in my thesis defense and for their insightful comments on the earlier version of the manuscript. I would like to thank Joanna Erfani and Corinne Lebet for their administrative assistance in the lab and the doctoral program. Also, many thanks to the European Research Council for funding this thesis through the Skill Acquisition in Human and Robots (SAHR) project.

I feel fortunate to have had such brilliant colleagues at LASA. I thank former and current LASA members for creating a positive atmosphere in the lab. I would like to express thanks to Sina Mirrazavi, Nadia Figueroa, Mahdi Khoramshahi, Iason Batzianoulis, Salman Faraji, and Konstantinos Chatzilygeroudis for their valuable guidance and help during my Ph.D. Many thanks to Salman, whose clever ideas and deep insight into real robotic challenges were great sources of inspiration for me. Special thanks to Iason, who was always available to help, listen to my problems, and give advice. I had the chance to collaborate with him, and I enjoyed his passion for and dedication to the research. I am highly grateful to Konstantinos, who has been my role model as a perfect researcher, a decent scientist, and an incredible programmer. We worked together on several projects, I have learned much from him, and he has, in many ways, contributed to my progress and research.

During Ph.D., I had the opportunity to collaborate closely with wonderful individuals. Big thanks to my collaborators and friends, Ilaria Lauzana, Sthithparagya Gupta, Kunpeng Yao, Xiao Gao, and Vincent Mendez. I am deeply thankful to my artist friend, Kunpeng, for his endless support and sharing experiences throughout our Ph.D. I must also thank Alice Guntli, Raphael Uebersax, Bruno Agostinho Da Costa, Sascha Frey, and Alexis Philip George-Georganopoulos, for their contributions to the study and for trusting me to supervise their

semester projects. I want to thank Michael Bombile for his wise pieces of advice during these four years and Jacob Hernandez for sharing good moments. I should thank our brilliant young researcher, Yang Liu, for being the best officemate one could ever have. I am also thankful to my talented friend, Harshit Khurana, for his tremendous help in revising my publications. Harshit is capable of conducting high-quality research, and I wish him a successful Ph.D.

I would like to thank four exceptional people: Mikhail Koptev, Carolina Correia, Walid Amanhoud, and Bernardo Fichera, who were by me all the time during the ups and downs of the Ph.D. I feel very fortunate to be their friend; and they are high value to me. Thank you, Mikhail, for being such a generous and honest friend. Thank you, Carol, for your unconditional support, kind heart, and all sweet memories. Thank you, Walid and Bernardo, for everything; you are the brothers I never had and always inspire me to be a better person.

I am also grateful to all my Iranian friends. I appreciate their help, support, and friendship. Special thanks to Mohammad Hossein Bahari and Melika Bahjat for their amazing personality and all trips and hikes that we did together.

Lastly, my heart is overwhelmed with gratitude to my family in Iran and my lovely Tara here in Switzerland. They are all that I have and those that make my life meaningful. After all the hardships of life, knowing that I have them gives me the strength to keep going. Thank you very much, Tara, for all the sacrifices you made and for being the remedy of all my stress.

Lausanne, October 24, 2022 Farshad Khadivar

Abstract

As technology continues to evolve, robots are becoming an intrinsic part of our lives. While robots surpass human capabilities in precision and speed, they are far from matching humans' ability to adapt to unexpected changes. Humans can quickly and compliantly respond to uncertainties and perform complex tasks such as dexterous manipulation. Robots in human-centric environments need to be equipped with human-like capabilities, primarily when such a machine interacts with us or the objects around us. Therefore, the thesis aimed at (i) obtaining a precise controller with human-like compliant capabilities and, with such a controller, (ii) designing a control pipeline to perform dexterous manipulation.

Learning, assessing, and constantly updating the underlying robot's dynamics is the first step to obtaining a precise and compliant controller. Accordingly, we present methods to learn and update dynamics models in the first two parts of the thesis. The first part of the thesis investigates supervised machine learning techniques for learning the inverse dynamic model of a robot prior to task execution. We propose a method for incrementally exploring a robot's configuration space and maximizing the information of the collected data.

In the second part, we focus on assessing and updating dynamic models during and after task execution. We offer a method that augments episodic model updates with online adaptation. We propose combining traditional model-based controllers with a learned residual inverse dynamics model. Then, we introduce an adaptive control law that adjusts the control reference online to account for model uncertainties and unforeseen disturbances.

We dedicate the third part of the thesis to developing a compliant control pipeline to achieve human-like dexterity tasks. We introduce a new robust and synchronized planning schematic for grasping and manipulating tasks. Our approach is to control and synchronize fingers based on dynamical systems. We combine our adaptive controller with joints' impedance regulation to guarantee high tracking accuracy and adapt to dynamic changes. We showcase that in conjunction with learning from human demonstration, our controller provides a robust solution for more complicated manipulations such as finger gaiting.

Lastly, we use the developed robotic hand controller in two applications in the human-centric environment. The task in the first application is to increase the dexterity of robotic prosthetic hands (RPHs) for individuals with a hand amputation using a new teleoperated-control scheme via electromyography. In the second application, we perform tactile surface exploration of apriori unknown objects through a novel informative path planning exploration

strategy.

In summary, this thesis offers a robot controller, compliant in interaction and faithful in tracking, for dexterous grasp and manipulation tasks. Results indicate substantial accuracy improvement over traditional approaches when using our incremental method for configuration exploration. We confirmed by various experiments that our residual model learning procedure could learn unmodeled dynamics in a hand full of trials and adapt online to perturbations in unpredictable environments. Finally, with extensive experiments and two applications, we showed that introducing a coordinated multi-finger system to our controller provides a robust solution for grasping and manipulating problems in uncertain environments.

Keywords: Compliant Torque Control, Learning Residual Dynamics, Robust/Adaptive Control, Dexterous Manipulation, Coupled Dynamical Systems, Tactile Exploration.

Résumé

À mesure que la technologie continue d'évoluer, les robots deviennent une partie intrinsèque de nos vies. Si les robots surpassent les capacités humaines en termes de précision et de vitesse, ils sont loin d'égaler la capacité des humains à s'adapter à des changements inattendus. Les êtres humains peuvent réagir rapidement et de manière conforme aux incertitudes et effectuer des tâches complexes telles que la manipulation agile. Les robots dans les environnements centrés sur l'homme doivent être équipés de capacités semblables à celles de l'homme, principalement lorsqu'une telle machine interagit avec nous ou les objets qui nous entourent. Par conséquent, la thèse vise à (i) obtenir un contrôleur précis avec des capacités d'adaptation similaires à celle de l'homme et, avec un tel contrôleur, (ii) concevoir un pipeline de contrôle pour effectuer une manipulation agile.

Apprendre, évaluer et mettre à jour constamment la dynamique sous-jacente du robot est la première étape pour obtenir un contrôleur précis et adaptatif. En conséquence, dans les deux premières parties de la thèse, nous présentons des méthodes pour apprendre et mettre à jour les modèles dynamiques. La première partie de la thèse étudie les techniques d'apprentissage automatique supervisé pour apprendre le modèle dynamique inverse d'un robot avant l'exécution de la tâche. Nous proposons une méthode pour explorer de manière incrémentale l'espace de configuration d'un robot et maximiser l'information des données collectées.

Dans la deuxième partie, nous nous concentrons sur l'évaluation et la mise à jour des modèles dynamiques pendant et après l'exécution de la tâche. Nous proposons une méthode qui augmente les mises à jour épisodiques du modèle avec une adaptation en ligne. Nous proposons de combiner les contrôleurs traditionnels basés sur des modèles avec un modèle de dynamique inverse résiduel appris. Ensuite, nous introduisons une loi de contrôle adaptatif qui ajuste la référence de contrôle en ligne pour tenir compte des incertitudes du modèle et des perturbations imprévues.

Nous consacrons la troisième partie de la thèse au développement d'un pipeline de contrôle adaptatif pour réaliser des tâches de dextérité de type humain. Nous introduisons un nouveau schéma de planification robuste et synchronisé pour les tâches de préhension et de manipulation. Notre approche consiste à contrôler et synchroniser les doigts sur la base de systèmes dynamiques. Nous combinons notre contrôleur adaptatif avec la régulation de l'impédance des articulations pour garantir une grande précision de suivi et s'adapter aux changements

dynamiques. Nous montrons qu'en conjonction avec l'apprentissage à partir de la démonstration humaine, notre contrôleur fournit une solution robuste pour des manipulations plus compliquées telles que la marche des doigts.

Enfin, nous utilisons le contrôleur de main robotique développé dans deux applications prenant place dans un environnement centré sur l'homme. Dans la première application, la tâche consiste à augmenter la dextérité des mains prothétiques robotiques (RPH) pour les personnes amputées de la main en utilisant un nouveau schéma de contrôle téléopéré via l'électromyographie. Dans la deuxième application, nous effectuons l'exploration tactile de la surface d'objets a priori inconnus par le biais d'une nouvelle stratégie d'exploration informative de planification de trajectoire.

En résumé, cette thèse propose un contrôleur de robot, adaptatif dans l'interaction et fidèle dans le suivi, pour des tâches de préhension et de manipulation agiles. Les résultats indiquent une amélioration substantielle de la précision par rapport aux approches traditionnelles lors de l'utilisation de notre méthode incrémentale pour l'exploration de la configuration. Nous avons confirmé par diverses expériences que notre procédure d'apprentissage de modèle résiduel pouvait apprendre des dynamiques non modélisées en une poignée d'essais et s'adapter en ligne aux perturbations dans des environnements imprévisibles. Enfin, grâce à des expériences approfondies et à deux applications, nous avons montré que l'introduction d'un système multi-doigts coordonné dans notre contrôleur fournit une solution robuste pour les problèmes de saisie et de manipulation dans des environnements incertains.

Zusammenfassung

Mit der fortschreitenden Entwicklung der Technologie werden Roboter zu einem festen Bestandteil unseres Lebens. Zwar übertreffen Roboter die menschlichen Fähigkeiten in Bezug auf Präzision und Geschwindigkeit, doch sind sie weit davon entfernt, sich an unerwartete Veränderungen anzupassen. Der Mensch kann schnell und flexibel auf Unwägbarkeiten reagieren und komplexe Aufgaben wie geschickte Manipulationen ausführen. Roboter in menschenzentrierten Umgebungen müssen mit menschenähnlichen Fähigkeiten ausgestattet sein, vor allem wenn eine solche Maschine mit uns oder den Objekten um uns herum interagiert. Ziel dieser Arbeit war es daher, (i) einen präzisen Controller mit menschenähnlichen Fähigkeiten zu entwickeln und mit einem solchen Controller (ii) eine Steuerungspipeline zur Durchführung geschickter Manipulationen zu entwerfen.

Das Erlernen, Bewerten und ständige Aktualisieren der zugrunde liegenden Roboterdynamik ist der erste Schritt zu einer präzisen und nachgiebigen Steuerung. Dementsprechend stellen wir in den ersten beiden Teilen der Arbeit Methoden zum Lernen und Aktualisieren von Dynamikmodellen vor. Im ersten Teil der Arbeit werden überwachte maschinelle Lernverfahren zum Erlernen des inversen dynamischen Modells eines Roboters vor der Ausführung einer Aufgabe untersucht. Wir schlagen eine Methode zur inkrementellen Erkundung des Konfigurationsraums eines Roboters und zur Maximierung der Informationen aus den gesammelten Daten vor.

Im zweiten Teil konzentrieren wir uns auf die Bewertung und Aktualisierung von dynamischen Modellen während und nach der Aufgabenausführung. Wir bieten eine Methode an, die episodische Modellaktualisierungen durch Online-Anpassung ergänzt. Wir schlagen vor, traditionelle modellbasierte Steuerungen mit einem gelernten inversen Dynamikmodell zu kombinieren. Dann führen wir ein adaptives Kontrollgesetz ein, das die Kontrollreferenz online anpasst, um Modellunsicherheiten und unvorhergesehene Störungen zu berücksichtigen.

Der dritte Teil der Arbeit widmet sich der Entwicklung einer nachgiebigen Steuerungspipeline, um menschenähnliche Geschicklichkeitsaufgaben zu erreichen. Wir stellen ein neues robustes und synchronisiertes Planungsschema für Greif- und Manipulationsaufgaben vor. Unser Ansatz besteht darin, die Finger auf der Grundlage dynamischer Systeme zu steuern und zu synchronisieren. Wir kombinieren unseren adaptiven Controller mit der Impedanzregulierung der Gelenke, um eine hohe Verfolgungsgenauigkeit zu gewährleisten und sich an dynamische

Veränderungen anzupassen. Wir zeigen, dass unsere Steuerung in Verbindung mit dem Lernen aus menschlichen Demonstrationen eine robuste Lösung für kompliziertere Manipulationen wie das Gehen der Finger bietet.

Schließlich setzen wir die entwickelte Roboterhandsteuerung in zwei Anwendungen in der menschenzentrierten Umgebung ein. In der ersten Anwendung geht es darum, die Geschicklichkeit von Roboterhandprothesen (RPHs) für Menschen mit einer Handamputation zu erhöhen, indem ein neues teleoperatives Steuerungsschema über Elektromyographie eingesetzt wird. In der zweiten Anwendung führen wir eine taktile Oberflächenerkundung von im Vorfeld unbekannten Objekten durch eine neuartige informative Pfadplanungsstrategie durch.

Zusammenfassend lässt sich sagen, dass diese Arbeit eine Robotersteuerung für geschickte Greif- und Manipulationsaufgaben entwickelt hat, die in der Interaktion nachgiebig und in der Verfolgung zuverlässig ist. Die Ergebnisse zeigen, dass unsere inkrementelle Methode zur Konfigurationsexploration die Genauigkeit im Vergleich zu traditionellen Ansätzen erheblich verbessert. Wir haben in verschiedenen Experimenten bestätigt, dass unser Residualmodell-Lernverfahren unmodellierte Dynamik in einer Hand voller Versuche erlernen und sich online an Störungen in unvorhersehbaren Umgebungen anpassen kann. Schließlich haben wir mit umfangreichen Experimenten und zwei Anwendungen gezeigt, dass die Einführung eines koordinierten Mehrfingersystems in unseren Controller eine robuste Lösung für Greif- und Manipulationsprobleme in unsicheren Umgebungen bietet.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Advances in modern robots have proved that repetitive tasks in industrial assembly lines or tedious household activities are a better fit for automated devices than humans. Robots in the industry tirelessly and reliably accomplish tasks with high precision and speed (Figure 1.1). Collaborative scenarios are other settings where robots are hired to ease tasks and improve the work quality. For robots to cooperate with humans, they not only have to be programmed to perfect their task execution but also need to be configured for safe interactions. Although many of the household activities or the tasks found in industrial workspace seem trivial for humans, they are notably complex for robots to perform. Robotics and Artificial Intelligence (AI) experts have dedicated considerable efforts to advancing robots that can accomplish various services in human-centric environments. In addition to safety requirements, the unpredictability of the human environment and the lack of dexterity in robots are among the critical restricting factors of their presence in our homes. Consequently, most commercially successful domestic and collaborative robots are limited to specific applications with minimal physical interaction between humans and robots (Figure 1.2).

The physical interaction protocol is the critical component distinguishing robots operating in an industrial setup from those working around humans. Robots close to humans must remain compliant to interaction forces, even though this is usually achieved at the cost of sacrificing precision (Buchli et al. (2009)). One way to alleviate the precision loss in compliant controllers is to improve our knowledge of the system's dynamics, i.e., the underlying dynamic model of the robot (Nguyen-Tuong and Peters (2011)) as well as the robot-environment interaction forces (Hogan and Buerger (2018)). The more exact our knowledge of the system's dynamics is, the more confident we are in predicting the robot's motions and interactive forces. Enhancing accuracy through updating the dynamics model should be an active and never-ending process, as these models have high nonlinearities and vary over time due to mechanical imperfection and changes in their environment. **Therefore, one solution to improve accuracy while remaining compliant in robot control is to *learn, assess,* and *update* our available models of the system's dynamics *before, during,* and *after* a task execution.**

(a) Collaborative manipulation[1]

(b) Scanning and fault detection[2]

(c) Repetitive pick and place[3]

(d) Precision welding[4]

(e) Warehouse automation[5]

Figure 1.1: Examples of robots in a structured environment: an environment that is known, predictable, or invariant.

In many modern robotic applications, the robot's surrounding environment is frequently subject to changes (e.g., gravity effect, light condition for vision) or new situations (such as a disturbance or facing an obstacle). For such applications, improving the model of dynamics or adjusting the controller parameters during task execution is imperative. Compared to a human-centric environment, the workspace of industrial robots is more structured – that is, it is known, predictable, and invariant. Low uncertainty in such environments allows stiff joints for robots, thereby delivering task precision (Dollar and Howe (2007)). On the other hand, robots in an unstructured environment are required to react and adapt to a host of uncertainties. Consider a human handing over objects to a robot and vice versa. In such a task, the object's shape and mass could vary, and each individual might pass the object differently. The robot must interpret human intention accurately to decide when to grasp/release the object. Once the object is grasped, the robot's job is to hold it securely even though the mass properties might differ from what was expected or modelled. **The robot, hence, must be capable of active model adaptation, coping with model uncertainties, and reacting to unforeseen situations.**

[1] https://blog.robotiq.com/whats-new-in-robotics-this-week-may-20
[2] https://new.abb.com/products/robotics/application-cells/3d-quality-inspection
[3] https://www.fanuc.eu/uk/en/robots
[4] https://www.robotics247.com/images/article/robot_welder.jpg
[5] https://www.logisticsmgmt.com/topic/category/warehouse


(a) Lawn mower[6]


(b) Vacuum cleaner[7]


(c) Segway robot[8]


(d) Telepresence-bot[9]

Figure 1.2: Examples of a robot in a human-centric environment

The necessity of compliant controllers is not restricted to unstructured environments. Even in industrial setups, compliant joint controllers are required for specific tasks. When multiple robots collaborate or interact with each other(e.g., grabbing and manipulating a box with two robotic arms), it is essential to avoid stiff joints as even a tiny error (in control or perception) puts the robots, the object, or the environment at high risk. **Although a compliant controller seems the natural choice for collaborative and manipulation tasks, it is not straightforward to couple multiple agents in a compliant and reactive fashion using traditional methods.** In such a scenario, a robot must follow a distributed reference motion when coordinating with other robots. Inaccuracies in task execution by one robot affect the reference motion; therefore, the influence of possible imprecision due to the need for compliant behavior may propagate throughout the system.

The human hand is a perfect example of practicing compliant coordination and interactions. For example, when screwing a light bulb (see Figure 1.3, right), we coordinate fingers to rotate the bulb while securing its position compliantly. Humans can coordinate different body parts and safely interact with the environment to perform a task. We use our hands as our predominant tools to interact with the outside world. The demand for developing dexterous skills has evolved the human hand into the most sophisticated body part with the highest degrees of freedom (DoFs) (Kontoudis et al. (2019)). We rely on our hands and their fine dexterity to perform everyday tasks, from basic manipulations, such as picking up a glass of water, to fine-manipulation skills, such as writing, knitting, and playing an instrument. In recent years, researchers have flexed their abilities to develop robotic hands that can emulate two main functionalities of the human hand (Melchiorri and Kaneko (2016)):

(i) *prehension*: the ability to grasp and hold objects of different sizes and shapes, and

(ii) *apprehension*: the ability to understand through active touch.

[6]https://myrobotmower.com
[7]https://www.irobot.ch
[8]https://www.theguardian.com/technology/2016
[9]https://assets.rbl.ms/25573765/origin.jpg

Figure 1.3: Practicing human like dexterity with a robotic hand

From a robotic perspective, human hands are versatile end-effectors that can grasp an object stably and manipulate it to the desired pose by applying forces at contact points. These end-effectors can also explore an unknown object and provide information via the sense of touch (Sommer and Billard (2016)). Here, we seek the same dexterous capabilities for a robotic hand to expand its applicability in a human-centric environment. **The motivation of this thesis is achieving robot controllers suited for the unpredictability of the human environment and capable of similar dexterous functionalities as the human hand.**

1.2 Thesis Objectives

In this thesis, we investigate ways of advancing robots' functionalities in human-centric environments using compliant controllers. We set our mission to develop control paradigms that equip robots with human-like capabilities, especially when such a machine interacts with us or with objects around us. Hence, **the thesis main objectives are:**

(i) obtaining a compliant and accurate robot control, and

(ii) advancing such controllers for performing human-like dexterous tasks.

To realize these objectives, we take inspiration from humans' remarkable ability to devise ways of learning, adapting, and mastering new manipulation tasks. With this in mind, **robots around humans must fulfill the following requirements:**

(i) Being able to learn the underlying dynamics before, during, and after task execution: In Section 1.3.1, we discuss the challenges within this requirement and our solutions for them. We achieve this goal in Chapters 3 and 4. First, we investigate supervised machine learning techniques for learning the robot's inverse dynamics (ID) model, which is instrumental in precise robot control and a key component in compliant manipulation. We propose a method to incrementally explore the configuration space of a robot and

maximize the information of the collected data for model learning. Next, we offer an approach that allows episodic model updates for the learned ID model, thereby obtaining higher precision and compensating for inevitable model mismatches.

(ii) Adapting to uncertainties and unforeseen situations while remaining compliant to reactive forces: In Section 1.3.2, we explain the challenges of obtaining such a control framework and our approach to address them. We realize this objective in Chapters 4 and 5. We introduce an adaptive control law that adjusts the model online to account for model uncertainties and unforeseen disturbances. Our adaptation framework comprises reactive planning (Chapter 4) and low-level adaptation (Chapter 5).

(iii) Using controllers that are programmable for performing dexterous tasks efficiently without structural changes in the environment: This objective is discussed in Section 1.3.3 and it is achieved in Chapters 5 and 6. In the second half of the thesis, we extend the developed controller in the first half for robust object grasping and manipulating with a robotic hand. First, we introduce a new adaptive and synchronized planning schematic that controls and synchronizes fingers based on dynamical systems. Then, we showcase the effectiveness of our robotic hand controller in two applications in a human-centric environment: one for prehension and the other one for apprehension.

1.3 Challenges and Solutions

This section details the challenges studied and approaches taken in this thesis.

1.3.1 System Dynamics, Modeling and Learning

Learning Dynamics

Learning the ID is an active research topic in the field of learning control. Indeed, an accurate model of the robot dynamics can be exploited to predict proper feedforward joint torques needed to achieve the desired trajectory. This avoids the use of high control gains in feedback (e.g., PID with large gains), which are often subject to instabilities. The more accurate the model is, the lower can be the feedback gains, allowing more room for compliance and safety when interacting with humans or the environment (Ko and Fox (2008); Burdet and Codourey (1998a); Nguyen-Tuong et al. (2008b)). Considerable efforts were dedicated to learning the ID models (Nguyen-Tuong et al. (2009); Nguyen-Tuong and Peters (2010); Meier et al. (2014, 2016)). However, acquiring such a model is challenging, not only due to unmodeled nonlinearities such as joint friction but also from a machine learning perspective (e.g., input space dimension, amount of data needed).

Similar to any learning framework, the efficacy of the learned ID model substantially depends on the training data supplied to it (Hitzler et al. (2019)). The data collected for learning should be rich enough to cover the robot's state space diversely. In other words, the sampling

Figure 1.4: Different examples of robot configuration during exploration of the KUKA LBR iiwa 14 joint space when using our configuration exploration approach.

trajectory that the robot should follow (while data is being recorded) needs span the robot's configuration space (see Figure 1.4) as comprehensively as possible. We approach this problem by focusing on generating sampling trajectories that efficiently explore the robot's joint space. We propose a method that incrementally generates the richest data based on information theory (Taneja (1989)) and probabilistic approaches (Murphy (2012)). Also, we offer a new path planning method based on inducing stable limit cycles in the phase space of each robot joint. This planning scheme controls the richness of the collected data in joint positions and velocities.

Episodic Model Updates

In many real-world scenarios, the model at hand and the reality do not match even if an accurate model is previously learned. Also, retraining the model from scratch for a new scenario is inefficient and costly. For instance, consider retraining the object-robot model every time a robot needs to grasp various objects. The efficiency of most of the model-based controllers is based on a relatively strong assumption: model precision, that is, the model adequately captures the underlying robot/environment model. Model-based controllers rely on high-gain PID regulators (Nakanishi et al. (2008)) to compensate for unmodeled dynamics or model imperfection. Even when the system is coupled with accurate state estimators and high-gain PID feedback, real-world experiments are subject to frequent failures due to model imperfections, friction, and actuator nonlinearities (Nguyen-Tuong and Peters (2011)).

On the other hand, humans and animals learn by trial and error. They learn how to perform new tasks or adapt to unforeseen situations. We envision a robotic system that similarly learns by trial-and-error: it tries to achieve the task, fails (e.g., the box slips from the hands of the robot), and tries again until it manages to realize the goal. For us, this episodic model update is only helpful if it is fast; imagine having to wait two days for a robot to learn how to perform a task while in a search and rescue scenario. We take Quadratic Programming (QP)-based controller as one example of model-based controllers that allow many robotic systems such as humanoids to successfully undertake complex motions and interactions and carefully coordinate a large number of degrees of freedom. We formulate an episodic residual ID model learning procedure to improve the model of the QP and show that it applies to a variety of robots with different dynamics. Then, we introduce a novel QP-based control scheme that can overcome model inaccuracies and significant model mismatches. The proposed control pipeline combines the slow ID model learning with a fast online adaptive control law in the taskspace to regulate the cost function of the QP.

1.3.2 Online Adaptation to Uncertainties

Episodic learning procedure by trial and error can fully capture model mismatches. However, we can update the ID model once the episode is over. In other words, the updated model only benefits the next episodes. In many situations, it is desired that the robot is capable of reacting to uncertainties online. In terms of learning nonlinearities, online adaptation is not as powerful as episodic learning; nevertheless, it delivers adequate flexibility to adapt to subtle changes among different tasks. Adaptive controllers (Ioannou and Sun (2012)) can compensate for such model inaccuracies and unmodeled dynamics. The adaptive nature of the controller mitigates the challenge of gain tuning for various desired behaviors (Ioannou and Sun (2012); Sun et al. (2017)). Furthermore, these controllers do not discard the valuable information from the previous trials. We seek to design a powerful control pipeline that combines the advantages of episodic model learning with fast online adaptation.

When controlling a robot compliantly, adapting to uncertainties becomes nontrivial as most adaptation techniques (Ortega and Spong (1989); Goodwin and Sin (2014)) are governed by only reducing the error in task execution, leading to stiffer joints. A naive solution would be to set bounds on the adaptation gains, frequently resulting in the well-known issue of gain saturation (Tao (2003)). In this thesis, to perform online adaptation in a compliant fashion, we follow two approaches: (i) outsourcing the adaptation by executing it only in the reference input of the model-based torque controller and (ii) regulating joint impedances to a set of desired values in conjunction with joint torques. We use the latter when an interaction is inevitable (e.g., grasping an object with unknown physical properties), which we explain in the next section. To achieve the former, we employ a model reference adaptive control (MRAC) for online reference adaptation in QP-based controllers to react online to nonlinearities and model uncertainties. MRAC methods have demonstrated substantial performance in addressing model uncertainties within the adaptive class of controllers, thereby emerging in

numerous studies (Azimi et al. (2019); Sharifi et al. (2021)). Also, this controller has been proven efficient for collaborative tasks. For instance, Culbertson et al. (2021) use a decentralized MRAC to control a group of collaborative robots in order to manipulate an object with unknown dynamic and geometrical properties.

1.3.3 Coordination and Human Level Dexterity

Using robotic hands in unstructured environments requires practicing human-level dexterity for both grasp and manipulation. Performing dexterous tasks, e.g., tilting a glass of water or screwing a lightbulb (Figure 1.3, left), though trivial for humans, is notably convoluted for robots to accomplish. In order to acquire a merely desirable performance for dexterous tasks, we must consider a collection of factors, such as morphological features, sensory equipment, large DoFs, control algorithms, and task planning strategies. Besides, in specific applications, other elements could exacerbate the existing challenges, e.g., the effects of the object's dynamics on object-robot interaction forces when manipulating the object within the hand. Accordingly, robotic hands can highly benefit from compliant controllers that are accurate in tracking, robust to uncertainties, and adaptive to unexpected conditions.

Securely grasping the object is an essential ingredient required prior to in-hand manipulation (Feix et al. (2016)); nevertheless, the main manipulation challenges arise after the object is grasped. The hand must carefully re-position the object by synchronously modulating the object-finger contacts/interactions (Prattichizzo and Trinkle (2016)). Robustly performing this is never easy and becomes even more complex in the presence of uncertain and inaccurate estimation of the system's mechanical properties (Okamura et al. (2000)).

We address the problem of ensuring robust in-hand manipulation when faced with a poor model of the object's dynamics. We take advantage of a combination of model-based and learning approaches that enables us (a) to estimate the dynamics of the object-finger forces, which might not be easy to determine, and (b) to control explicitly for different perturbations, e.g., changes in the position of the object. We propose a method based on coupled dynamical systems to synchronize the fingers' dynamics robustly. We also offer a novel control pipeline using an adaptive torque controller to face model uncertainties. This controller provides a live adaptation of the position and force generated by the fingers to stabilize the object.

For an application illustration, consider the example of tilting a glass half-filled with water in Figure 1.5. Here, moving fingers in coordination is crucial. Suppose one finger moves faster than the others, i.e., with asynchronous displacement. In that case, the water in the glass moves faster, and the object gains extra momentum, thus rendering the grasp hard to control. Furthermore, adaptation here is equally critical. As the liquid spreads in the glass, the hand must re-balance the force in order to compensate for the changes in the mass distribution. Assume further that the hand dynamics is only partially known. Consequently, the controller must compensate for its dynamic uncertainties and prevent the object from veering away from the desired trajectory.

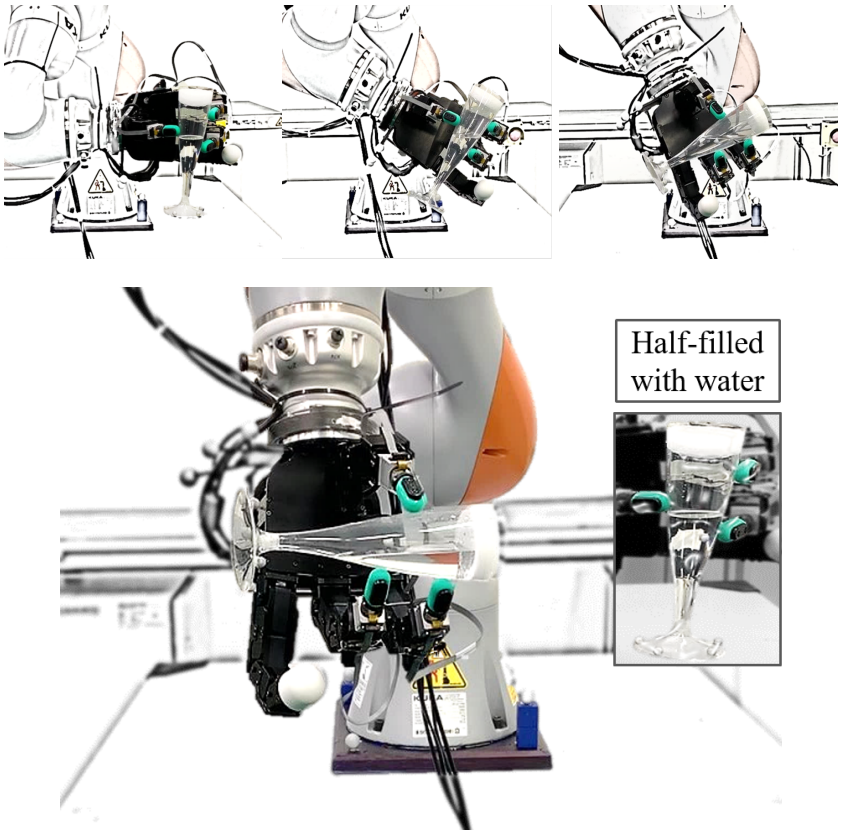


Figure 1.5: Rotating a half-full champagne glass, an object with varying mass distribution. The glass is first grasped in a vertical pose (top left) then rotated to reach a stable horizontal pose (central). The glass is stabilized by optimizing the object-level grasp-impedance and by extracting contact frames from tactile sensors at contact points. Contact forces of all fingers are synchronously modulated based on a coupled dynamical system and an adaptive control scheme.

1.3.4 Compliant Robotic Hand Controller in Human-Centric Environments

Application I: Prehension

The human hand owes its dexterity to its individuated control of finger motion and precise sense of touch, and to dedicated brain processes for ensuring fluent control of objects once in hand (Kontoudis et al. (2019); Castiello (2005)). Being deprived of this dexterity following an amputation affects one's life considerably. Even the simplest chores, such as placing the cap on a bottle or inserting a spoon in a cup, become arduous if not impossible. The lack of touch perception and the absence of fine finger motions are the predominant hindering factors that render reorienting or repositioning an object impossible. If one could restore some, even if a tenuous part, of this dexterity through the help of automation and robotics, this would have immediate benefits for people with an amputation.

While robotics has made vast progress in controlling human-like robotic hands (Melchiorri and Kaneko (2016); Okamura et al. (2000)), these advancements have rarely been used to control prostheses. The main reasons are that (i) robotic prosthetic hands (RPHs) remain under-actuated with simple position or speed control, and (ii) we still lack robust decoding strategies to infer and translate the user's intentions into fine and individuated finger motions of an RPH. The latter is a bottleneck that reduces the incentive to develop more complex robotic hands for people with trans-radial amputation.

Shared-control approaches that give some authority to an autonomous controller on-board the prosthesis is an alternative to direct Electromyography (EMG) control. This shared-control scheme may improve the dexterity and versatility of RPHs for people with trans-radial amputation. We use the developed controller in this thesis to enable human-like control of the fingers individually or coordinatedly. This controller enables online adaptation of the positioning of fingers and stabilizing the object in hand through forces applied by the fingers. Such enhancements permit the execution of robust grasps with multi-finger robotic hands. The shared control approach allows subjects to perform a continuum of manipulation: from grasping an object to manipulating it in air and during insertion when in contact with another object.

Application II: Apprehension

Haptic feedback and tactile perception are among the means by which robots interact with the environment or humans. In some cases, a robot can identify an object with precision only with tactile perception (e.g. the object is inside a box among other objects). The problem of autonomously exploring a priori unknown objects is still remarkably challenging in robotics. It becomes even more complicated when the object shape is complex, like having a handle, narrow edge, or a hole, or when the environment is uncertain (there are multiple non-fixed objects). Tactile perception can only provide local information; thus, gathering and interpreting sensory information is an active process of exploitation and exploration. Collecting tactile data on an object's shape actively and efficiently is the main focus of this part of the thesis. We propose novel solutions for exploration and exploitation problems while employing an adaptive compliant controller for the robotic hand.

We propose novel solutions for exploration and exploitation problems while employing our developed adaptive compliant controller for the robotic hand. We separate Gaussian Process (GP) model used for path query from GP that models the implicit surface. Since exploration is myopic, we construct a GP model from the local data at each query point. We propose a dynamic approach for the evolution of parameters in the path query GP. This method takes the connectivity of points and local densities and changes parameters to dynamically increase the entropy of the collected data and the efficiency of exploration. This way, we avoid the issues with fixed hyperparameters and, simultaneously, acquire a more informative and balanced sample data set from the object surface.

1.4 Summary of Contributions and Thesis Outline

The principal outcome of this thesis is a robot controller for dexterous grasp and manipulation that is both compliant in interaction and accurate in tracking. Here, we overview the structure and the core contributions of the thesis in each chapter. This section is summarized in Figure 1.7. The link to supplementary materials and multimedia of each chapter is listed in Figure 1.6.

Chapter 2

This chapter provides a background summary and reviews the preliminary materials needed to follow the developed approaches in the thesis.

Chapter 3

With the method developed in this chapter, we obtain an information-rich dataset for learning the inverse dynamics of a serial robotic manipulator. Using supervised machine learning techniques, we contribute by:

(i) proposing a novel method, called Max-Information Configuration Exploration (MICE), that incrementally explores the robot's configuration space and generates information-rich data while computing parameters of a trajectory set online; and

(ii) introducing a new set of excitation trajectories that explores the robot's configuration through imposed stable limit cycles in robot joints' phase space while satisfying feasibility constraints and physical bounds.

We benchmark MICE against state of the art in terms of data quality and learning accuracy. The proposed methodology for data collection, model learning, and evaluation is validated with a KUKA LBR iiwa 14 robotic arm, where the results prove significant improvement over traditional approaches. This study is published in (Khadivar et al. (2021a)).

Chapter 4

In this chapter, we propose to augment (a) traditional model-based controllers (QP-based control, for instance) with a learned residual inverse dynamics model and (b) an adaptive control law that adjusts the reference signal of the model-based controller online to account for model uncertainties and unforeseen disturbances. In particular, we propose

(i) An episodic procedure of learning residual ID model for QP-based control where we use an expressive and differentiable Gaussian Process with Rigid Body Dynamic (RBD) model as prior. In our method, ID is differentiable with respect to the optimization

variables; hence, it actively exploits the learned ID model within the optimization of the QP-based control.

(ii) A novel scheme for continuous online reference adaptation in the cost function QP. To this end, we employ an adaptive controller that avoids manual tuning, addresses model uncertainties online, and achieves a faster convergence for ID model learning.

We extensively evaluate our method in simulation on several robotic scenarios ranging from a 7-DoFs manipulator tracking a trajectory to a humanoid robot performing a waving motion for which the model used by the controller and the one used in the simulated world do not match (unmodeled dynamics). Finally, we validate our approach on physical robotic scenarios where a 7-DoFs robotic arm performs tasks where the model of the environment (exact mass, friction coefficients, etc.) is not fully known. Our results showcase that our method (i) is able to compensate for large model inaccuracies in little interaction time (i.e., in a few trials) and (ii) consistently outperforms the baselines. At the time of writing, the work presented in this chapter is under the second round of review in a robotic journal.

Chapter 5

In this chapter, we present a control strategy based on coupled dynamical systems, whereby the fingers move in synchronization using an intermediate dynamics responsible for coordinating fingers. To adapt to changes in forces due to model uncertainties and unexpected disturbances, we employ an adaptive torque-controller combined with a joint impedance regulator that guarantees high tracking accuracy while adapting to dynamic changes. We validate the approach in multiple experiments on a 16-DoFs robotic hand grasping and manipulating objects with different mass properties, e.g., uneven or varying mass distribution in a glass half-filled with water. We show that the robot can compensate for disturbances generated by internal dynamics and external perturbations. Additionally, we showcase how our controller, combined with learning from human demonstration, provides a robust solution for more complicated manipulations such as finger gaiting. In this chapter our main contributions in this chapter are:

(i) introducing a novel intermediate dynamical system coupling method to achieve coordinated finger motions; and

(ii) tracking fingers' desired velocity using a joint-level adaptive torque controller. We regulate joint impedances to implement real-world grasp and manipulation tasks.

Additionally, we show that this approach can be combined with learning from human demonstration to learn a desired trajectory for the object and identify the corresponding sequence of desired motion for the fingers. We further show that learning can be used to identify roles for each group of fingers and embed this in the intermediary coupling. At the time of writing,

the work presented in this chapter is accepted in the journal of IEEE Transaction on Robotics (T-RO) and it is under publication process.

Chapter 6

This chapter comprises two applications of our compliant robotic hand controller in human-centric environments.

Application I: In this part of the thesis, we analyze teleoperated control schemes to increase the dexterity of robotic prosthetic hands for people with trans-radial amputation. We propose a novel shared control strategy for robust object manipulation that combines autonomous control of forces exerted by the robotic hand with EMG motion decoding. By delegating control of forces to the prosthesis' on-board control, one speeds up reaction time and improves the precision of force control. Such a shared-control mechanism may enable amputees to perform fine insertion tasks solely using their prosthetic hands. To this end, we propose:

(i) a virtual-object-based control that regulates online the interaction forces at contact points using tactile feedback and our autonomous and adaptive compliant controller; and

(ii) an interface that employs feedback-based state machines to integrate high-level and low-level robot control.

We conduct a 3-days long longitudinal study with eight healthy subjects tasked to control a 16-DoFs robotic hand to insert objects in boxes of various orientations. Results indicate that when combined with incremental EMG decoding, robotic assistance leads to significantly less failure and faster completion time in task execution. Training to use the assistive device is fast, and, in less than one day of practice, all subjects managed to control the robotic hand with high accuracy. At the time of writing, the work presented in this section is submitted to a robotic journal.

Application II: This part concerns the tactile surface exploration of a priori unknown objects with a robotic hand for online shape reconstruction. In essence:

(i) we propose an online exploration strategy that actively maximizes the entropy of the acquired data while dynamically balancing the global knowledge and the local complexity of the exploration; and

(ii) to allow for multi-contact exploration with a robotic hand, we offer an optimization-based planning algorithm that adapts the hand pose to the local surface geometry online and maximizes the kinematic properties of each finger during exploration.

We show that our method can efficiently explore objects with arbitrary shapes, e.g., having a handle, hole, or narrow edges. We benchmark our approach against state of the art in a simulated environment and showcase the strength of this method in learning object shapes with different complexities. At the time of writing, the work presented in this section is under preparation for submission to a robotic journal.

Chapter 7

In this chapter, we present a summary of this thesis, outline the key contributions, and discuss avenues for future work.

(a) Chapter 2 (b) Chapter 3 (c) Chapter 4

(d) Chapter 5 (e) Chapter 6.I (f) Chapter 6.II

Figure 1.6: Multimedia contents. Click on each caption or scan its QR code.

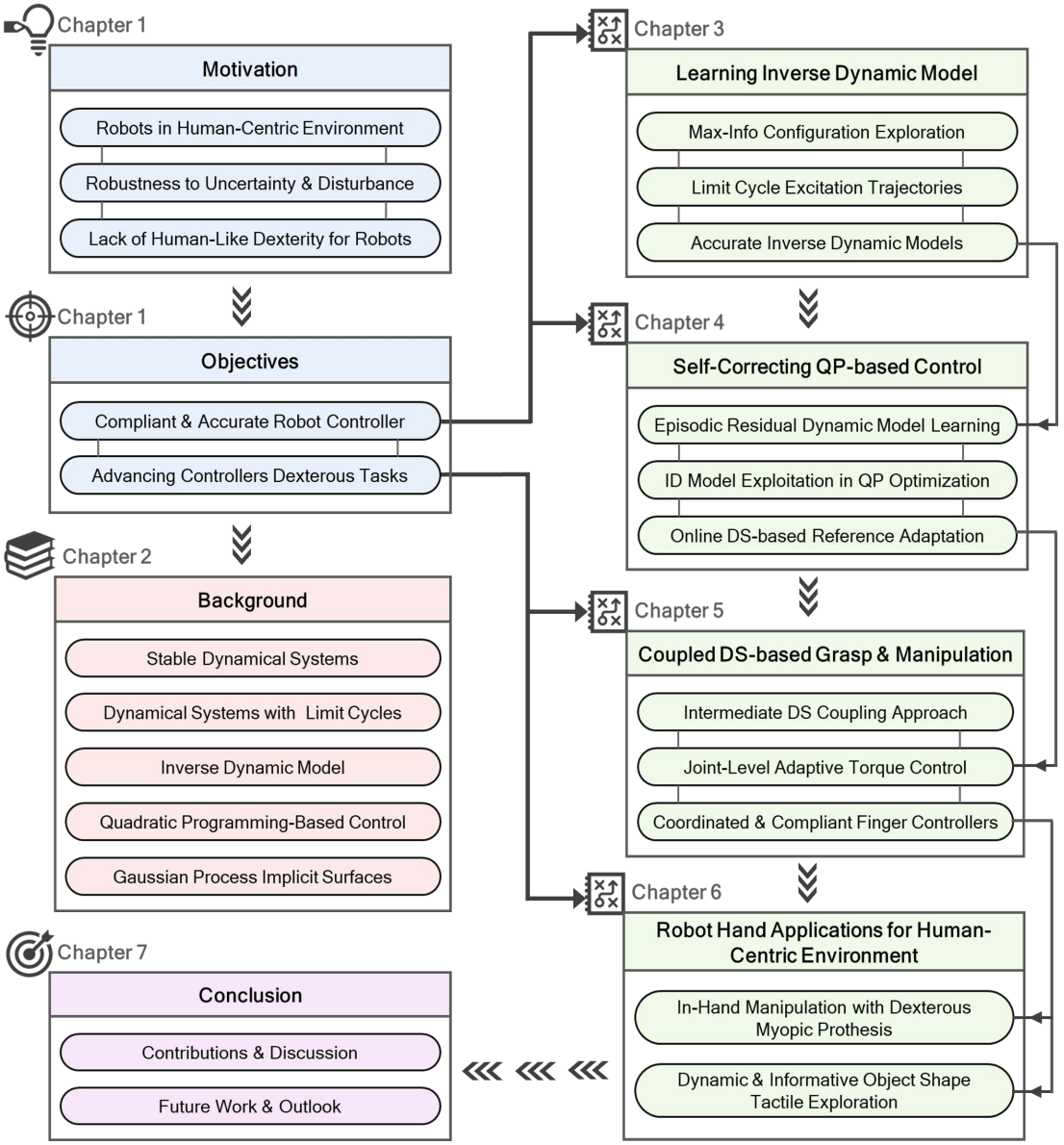


Figure 1.7: Thesis outline. Chapter 1 states the motivation, objectives, and a brief overview of the approaches of the thesis. Chapter 2 reviews the background and preliminary needed material for the developed methods in the following chapters. Chapter 3 and Chapter 4 focus on the first objective, whereas Chapter 5 and Chapter 6 concentrate on the second objective. In Chapter 7, the thesis is concluded by summarizing the key contributions and discussing avenues for future research.

2 Background

2.1 Stable Dynamical Systems

Dynamical System (DS) is a globally stable velocity field that provides the system's controller with desired velocities. A DS function $f(\boldsymbol{\xi})$ is continuous, autonomous, state-dependent, and converges to an *attractor* $\boldsymbol{\xi}^*$. This attractor can represent either a fixed state or a specific limit cycle; see Section 2.2. A DS function can also be expressed as a nonlinear combination of linear DSs to be:

$$f(\boldsymbol{\xi}) = \sum_{k=1}^{K} h^k(\boldsymbol{\xi})\left(\mathbf{A}^k\boldsymbol{\xi} + \boldsymbol{b}^k\right) \tag{2.1}$$

with $\boldsymbol{\xi} \in \mathbb{R}^n$ being the states of the system. $\mathbf{A}^k \in \mathbb{R}^{n\times n}$, as state matrix, and $\boldsymbol{b}^k \in \mathbb{R}^n$ construct the k-th linear system. The state-dependent mixing function, $h^k(\boldsymbol{\xi})$, represents a relative effect of the k-th system on shaping the overall nonlinear system. Given a desired robot motion, DS parameters $\mathbf{A}^k$, $\boldsymbol{b}^k$ and $h^k(\boldsymbol{\xi})$ can be either designed or learned from a demonstration (Khansari-Zadeh and Billard (2011)). Regardless of the acquisition method, these parameters must satisfy the following sufficient conditions for stability in the sense of Lyapunov :

$$\begin{cases} (\mathbf{A^k})^T\mathbf{P} + \mathbf{P}\mathbf{A^k} \prec -\mathbf{Q}^k \\ (\mathbf{Q}^k)^T = \mathbf{Q}^k \succ 0,\ (\mathbf{P}^k)^T = \mathbf{P}^k \succ 0 \\ \boldsymbol{b}^k = -\mathbf{A}^k\boldsymbol{\xi}^* \end{cases} \tag{2.2}$$

where $\mathbf{Q}^k \in \mathbb{R}^{n\times n}$ determines the convergence rate of k-th linear system to $\boldsymbol{\xi}^*$ and $\mathbf{P}^k \in \mathbb{R}^{n\times n}$ forms the potential function of the desired velocity field. DS-based controllers –i.e., controllers driven by a DS– are robust to perturbations because DS nests all possible trajectories to reach the target point, i.e., attractor.

2.2 Dynamical Systems with Limit Cycles

The material presented in this section is adopted from (Khadivar et al. (2021b)).

DS for controlling rhythmic patterns are distinct from DS used to control point to point motion. A cyclic motion can be embedded into a DS with a stable limit cycle (Ernesti et al. (2012)). Efforts to enable a single representation for periodic and non-periodic DS were offered in (Gams et al. (2015)), a framework to iteratively learn an initial discrete motion followed by a periodic one using a time-dependent DS. In (Khadivar et al. (2021b)), we offered an approach that encodes periodic and discrete dynamics in a single time-invariant DS by exploiting the concept of bifurcation, allowing for smooth transitions between the two. We introduced a form of such DS in both 2D and 3D spaces, and proposed a parameterization of the DS to offer an explicit way to control for its behavior, namely speed, location of the target and shape of the limit cycle. A DS with bifurcation can be represented by their normal forms as follows:

$$\mathbf{f}(\rho,\theta,\mu) = \begin{cases} \dot{\rho} = \rho(\rho^2 - \mu) \\ \dot{\theta} = \omega \end{cases} \tag{2.3}$$

with polar radius $\rho = \sqrt{x_1^2 + x_2^2}$ and polar angle $\theta = \tan(\frac{x_2}{x_1})$, and μ the bifurcation parameter, $\omega \in \mathbb{R}, \omega \neq 0$ an open parameter determining the frequency. This approach provides explicit control of the amplitude, the frequency and the goal of the movement which simplifies robot's motion modification and DS modulation in real-time for a user.

Limit Cycles in Two Dimensions

To formulate a DS with a limit cycle, we can simplify the system by converting the task coordinates $\mathbf{x} \in \mathbb{R}^2$ to polar coordinates $\mathbf{r} \in \mathbb{R}^2$. As simplification, we rename the bifurcation parameter as *target radius* ρ_0. The streamlines of this system generate parabolic motions that stabilize either an attractor point (here the origin of the system) or transform into ellipses at limit cycles, around the origin. The energy E of the system in polar coordinates is given as:

$$E(\dot{\rho},\dot{\theta},\rho) = \frac{N}{2}(\dot{\rho}^2 + \rho^2\dot{\theta}^2) + U(\rho,\rho_0) \tag{2.4}$$

where $\dot{\rho}$ and $\dot{\theta}$ are radial and angular velocities and N is constant inertia of the system. U is the potential function of system which is parametrized by the radius of the orbit. The energy E reaches its minimum on each stable limit cycle; therefore, we have:

$$\dot{E} = N(\dot{\rho}\ddot{\rho} + \rho\dot{\rho}\dot{\theta}^2 + \rho^2\dot{\theta}\ddot{\theta}) + dU(\rho,\rho_0)\dot{\rho} = 0. \tag{2.5}$$

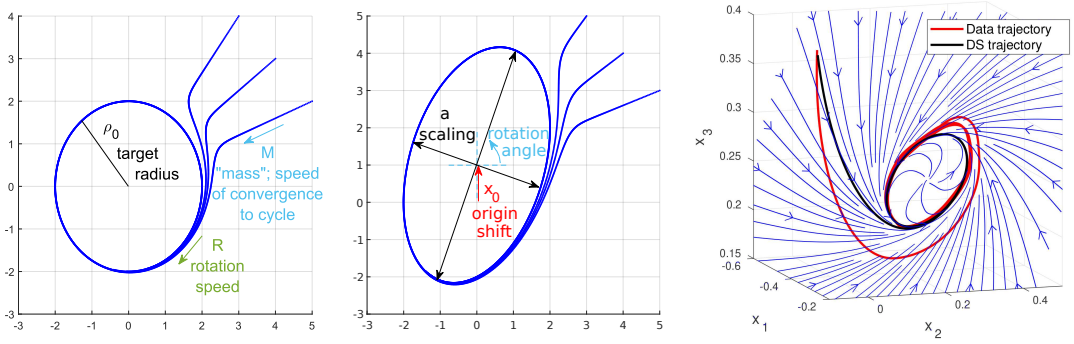


Figure 2.1: Schematics for target radius, “mass” and rotation speed (left), shifting, scaling and rotation (middle), and cyclic motion in Cartesian space (right).

By assuming that $\ddot{\rho} = -K_\rho \dot{\rho}$ with K_ρ being a positive gain, for Eq. 2.5 to hold, and considering $U(\rho, \rho_0)$: $U(\rho) = \frac{K_\rho}{2}(\rho - \rho_0)^2$,we derive the DS in polar coordinates with respect to ρ_0 as follows:

$$\mathbf{f}(\rho, \theta, \rho_0) = \begin{cases} \dot{\rho} = -\sqrt{M}(\rho - \rho_0) \\ \dot{\theta} = R\mathrm{e}^{-M^2(\rho-\rho_0)^2} \end{cases} \tag{2.6}$$

$M = \frac{1}{2N} > 0$ is a parameter that modulates the strength of the attraction and can be used to modulate the speed at which the robot moves. We can see from Eq. 2.6 that the differential equation of the polar radius vanishes at $\rho = \rho_0$. The differential equation of the polar angle θ, instead, is numerically different than 0 only in a neighborhood of ρ_0 which is defined by M. Therefore, the parameter M determines both the speed of convergence to the limit cycle and the distance at which the dynamics starts to rotate towards the stable ellipse. Finally, $R \in \mathbb{R}$ is an additional parameter that permits to easily change the angular speed (e.g. by doubling R, the speed will double), as well as to invert the rotation of dynamics around the limit cycle. Thus through modulation of M and R one can control the convergence rate and maximum velocity of the designed DS. Also when learning M and R from demonstration, the acquired DS captures the velocity limits of the demonstrated trajectories. Figure 2.1 shows these three parameters in an example of the introduced DS in 2D.

Extension to Three Dimensions

The formulated DS may be expanded to a 3-dimensional case by using spherical coordinates in place of polar ones. We can define the elevation angle ϕ, as the angle between x_3 and the plane formed by x_1 and x_2, and θ represents the azimuth angle (i.e. between x_1 and x_2). The additional angle's differential equation is defined as linear and vanishing in 0, such as in the 3D system:

$$\mathbf{f}(\rho, \phi, \theta, \rho_0) = \begin{cases} \dot{\rho} = -\sqrt{M}(\rho - \rho_0), \\ \dot{\phi} = -\sqrt{M}(\phi), \\ \dot{\theta} = R\mathrm{e}^{-M^2(\rho-\rho_0)^2}. \end{cases} \tag{2.7}$$

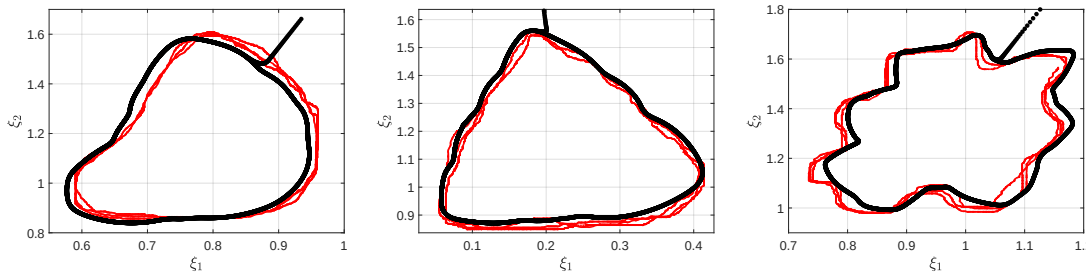


Figure 2.2: Examples of learning nonlinear limit cycles from demonstrations.

Extension to Nonlinear Limit Cycles

We can extend the proposed linear limit cycles (Eq. 2.6) to nonlinear form by using the concept of diffeomorphism. Diffeomorphisms have been used with a lot of success for learning and stabilizing nonlinear DS by learning a mapping to a simpler linear space (Perrin and Schlehuber-Caissier (2016); Neumann and Steil (2015); Rana et al. (2020)). The same principle cab be used to seek a mapping function $\boldsymbol{s}(\cdot)$ to send a nonlinear DS into a space where the DS is simpler to learn. DS (2.6) is our base oscillator, and we search a diffeomorphism from this base oscillator to the desired nonlinear limit cycle, and back.

For limit cycles and phase oscillators, scaling the polar radius depending on phase angle is sufficient to modify a limit cycle into a nonlinear one (Ajallooeian et al. (2013)). Therefore, we can assume that there exists a phase-based radial mapping function $\boldsymbol{s}(\theta)$ that scales the acquired linear DS in (2.6) to our desired nonlinear limit cycle:

$$\rho_n = \boldsymbol{s}(\theta)\rho_b \tag{2.8}$$

where ρ_b and ρ_n are the radius of base and nonlinear limit cycles, respectively. For better readability, we drop subscript n in ρ_n, and refer to $\boldsymbol{s}(\theta)$ as $\boldsymbol{s}$. Given this, if $\forall\theta : \boldsymbol{s} \neq 0$ and $\boldsymbol{s}$ is at least once differentiable, then $\boldsymbol{s}$ will define the Jacobian determinant of the desired diffeomorphism (Ajallooeian et al. (2013)). To form the nonlinear DS, we can take the derivative of Eq. 2.8:

$$\dot{\rho}(\rho,\theta) = \boldsymbol{s}\dot{\rho_b}\Big(\frac{\rho}{\boldsymbol{s}}\Big) + \frac{\rho}{\boldsymbol{s}}\frac{\partial \boldsymbol{s}}{\partial\theta}\dot{\theta}\Big(\frac{\rho}{\boldsymbol{s}}\Big). \tag{2.9}$$

Using Eq. 2.6 in Eq. 2.9, we obtain the following expression for the nonlinear limit cycle:

$$\begin{cases} \dot{\rho} = -\sqrt{M}\big(\rho - \boldsymbol{s}\rho_0\big) + \dfrac{R\rho}{\boldsymbol{s}}\dfrac{\partial \boldsymbol{s}}{\partial\theta}\mathrm{e}^{-M^2(\frac{\rho}{\boldsymbol{s}}-\rho_0)^2} \\ \dot{\theta} = R\mathrm{e}^{-M^2(\frac{\rho}{\boldsymbol{s}}-\rho_0)^2}. \end{cases} \tag{2.10}$$

Extracting $\boldsymbol{s}$ and real-time computation of $\partial\boldsymbol{s}/\partial\theta$ are the added complexities of Eq. 2.10 compared to Eq. 2.6. Figure 2.2 shows examples of nonlinear limit cycles learned from demonstration.

2.3 Inverse Dynamics Model

Learning the inverse dynamics is an active research topic in the field of learning control. The generalized dynamics of a N-DoF robotic arm can be modelled as (Spong and Vidyasagar (2008)):

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{2.11}$$

where $\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \in \mathbb{R}^N$ are the joint state positions, velocities and accelerations respectively. $\boldsymbol{\tau} \in \mathbb{R}^N$ is the joint torque vector, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{N\times N}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^N$ contains the Centripetal and Coriolis terms, and $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^N$ is the gravity torques. From the general rule of robotic chain (Hollerbach et al. (2016)), Eq. 2.11 can be transformed to:

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\boldsymbol{\beta} + \boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) \tag{2.12}$$

where $\boldsymbol{\Phi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\boldsymbol{\beta}$ represent the ideal Rigid Body Dynamics (RBD) of the robot. $\boldsymbol{\Phi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) \in \mathbb{R}^{N\times P}$ is the *Regressor* matrix incorporating the kinematic terms and $\boldsymbol{\beta} \in \mathbb{R}^P$ contains the physical parameters of the robot. The ideal RBD is linear in terms of $\boldsymbol{\beta}$ and non-linear with respect to the joint state vectors. The structured error-term $\boldsymbol{\epsilon}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ represents nonlinearities due to joint friction and unmodeled dynamics. Approaches to acquiring the robot's dynamics include:

(i) *RBD-based learning*: identifying the physical parameters, $\boldsymbol{\beta}$, of the model (Stürz et al. (2017); Hollerbach et al. (2016); Burdet and Codourey (1998b));

(ii) *RBD-free learning*: finding the mapping $\boldsymbol{\tau} = \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ with no prior knowledge on the model (Nguyen-Tuong et al. (2008b,a)); and

(iii) *RBD-residual learning*: compensating the error of RBD-based models by learning a mapping $\boldsymbol{\epsilon} = \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ as the *Residual Model* (Nguyen-Tuong and Peters (2010); Camoriano et al. (2016); Gijsberts and Metta (2012)).

RBD-based learning techniques utilize prior knowledge on RBD to improve joint torques' prediction by identifying the robot's physical parameters ($\boldsymbol{\beta}$) . Since these techniques learn the inverse dynamics using a predetermined model and work with a finite quantity of parameters, the data requirement complexity for training these models is the lowest as compared to the other techniques (Nguyen-Tuong et al. (2008b); Gribovskaya et al. (2011)). On the other hand, RBD-free learning techniques depend solely on data requirement, given the lack of prior and model. RBD-residual learning techniques, owing to their similarities with both RBD-based and RBD-free techniques for learning the inverse dynamics, have intermediate to high data requirements (Nguyen-Tuong and Peters (2010); Schölkopf et al. (1998); Kocijan et al. (2004)).

While offline learning frameworks work with a fixed dataset, online learning frameworks offer flexibility with regards to when (or where) the model training takes place. However, online frameworks require continuous excitation, which may not necessarily be present in the

incrementally supplied data (Nguyen-Tuong and Peters (2010)). Also, online learning can be used in tandem with a model first learned offline to incrementally improve the accuracy of the prediction (Meier et al. (2016)). For learning the inverse dynamics in this study, Chapter 3, we focus on offline learning frameworks.

2.4 Gaussian Process Implicit Surface Representation

Implicit surfaces are a popular choice to represent arbitrary shapes with complex topology as they are smooth and can incorporate object geometric information. An implicit surface is defined as a manifold $\mathscr{S} \subset \mathbb{R}^d$ that has a zero level of a scalar function $f : \mathbb{R}^d \mapsto \mathbb{R}$ (Williams and Fitzgibbon (2006)):

$$\mathscr{S} \triangleq \{\boldsymbol{x} \in \mathbb{R}^d | f(\boldsymbol{x}) = 0\} \tag{2.13}$$

the function f outputs $f = +1$, $f = -1$, and $f = 0$ for points outside, inside, and on the object surface respectively. Williams and Fitzgibbon (2006) proposed to model the implicit surface of f with Gaussian processes (GPs). Gaussian Process Implicit Surface (GPIS) shows high flexibility in modeling complex surfaces. It allows embedding prediction uncertainty in probabilistic ways, making GPIS a powerful tool for exploration purposes. The core of our multi-contact exploration strategy, Section 6.2, is based on this uncertainty measure from GPIS.

GPs are a collection of random variables, any finite number of which have a joint Gaussian distribution (Rasmussen (2003)). With GP, f can be defined as a distribution over functions: $f(\boldsymbol{x}) \backsim \mathscr{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$, where $m(\boldsymbol{x})$ is a prior mean function and $k(\boldsymbol{x}, \boldsymbol{x}') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a covariance function, also known as the kernel function. Given an observation set $\mathscr{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^M$, for a query point $\boldsymbol{x}_*$, the predictive distribution $f(\boldsymbol{x}_*)$ is extracted by conditioning the joint Gaussian prior distribution on the observation:

$$p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \backsim \mathscr{N}(\bar{f}_*, \mathbb{V}[f_*]) \tag{2.14}$$

with

$$\bar{f}_* = m(\boldsymbol{x}_*) + \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{X})) \tag{2.15}$$

and

$$\mathbb{V}[f_*] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{k}_* \tag{2.16}$$

$\boldsymbol{X}$ and $\boldsymbol{y}$ are the observed inputs and outputs. The components of matrix $\boldsymbol{K}(\cdot, \cdot)$ are computed by a defined kernel function: $k_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Vector $\boldsymbol{k}_* = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}_*)$ is the kernelized similarity of the query point $\boldsymbol{x}_*$ to the observed inputs.

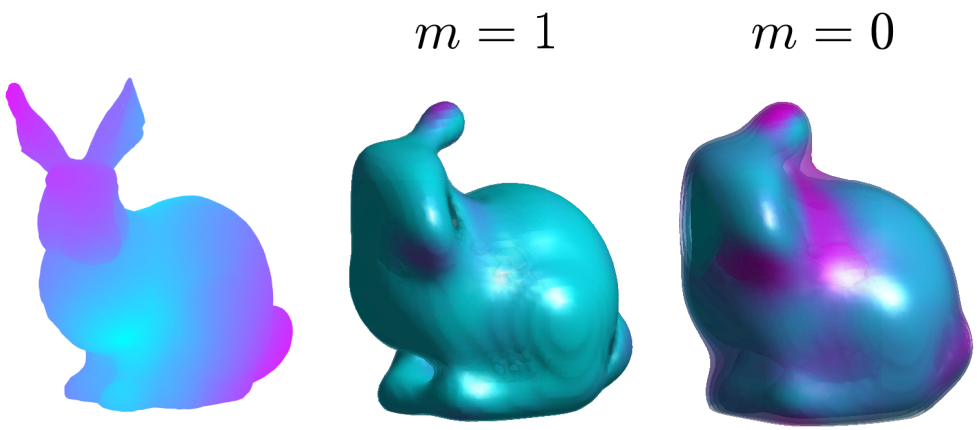


Figure 2.3: Object shape reconstruction from exploration data. Same sampled data (800 points) from ground truth is exploited to reconstruct the shape of the object (left) using GPIS with RBF kernel and prior $m = 1$ (middle), and $m = 0$ (right).

Prior Mean

Likewise (Dragiev et al. (2011); Driess et al. (2017)), we set $m = 1$ as the prior mean, meaning that any query point does not belong to the object surface unless there is a similarity to the observed points on the surface. Choosing $m = 0$, as in (Björkman et al. (2013); Yi et al. (2016)), causes shape distortion due to GP function tendency to fall back on prior, see Figure 2.3. Williams and Fitzgibbon (2006) proposed using the thin-plate spline kernel to avoid falling back to the mean function. Conversely, considering $m = 1$ renders falling back to mean prior a desirable behavior, allowing for more flexible and mathematically convenient kernels than thin-plate splines. In (Driess et al. (2017)), authors indicate that with $m = 1$ there is no need for recording off-surface points. On the contrary, we believe having off-surface points as observation is still required. In online exploration, we only access the Euclidean distance between points as the similarity measure; in other words, we do not know geodesic distance a priori. Therefore, off-surface points are still needed to improve the marginal likelihood of the shape's curve more accurately, which is crucial for tangent plane estimation. Besides, off-surface points affect the optimal range of kernel hyperparameters, and, unlike in simulation, they are easy to collect in real-world practices.

Kernels and Model Selection

From experience, we have observed that exploration performance (the decreasing rate of reconstruction error) becomes kernel indifferent by collecting more and more points. Thus, the choice of kernel matters most in the early phases of exploration. We opt for the RBF kernel as we observed that w.r.t. the number of data points, we obtain higher regularity in shape

prediction. The RBF kernel is formulated as follows:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_l^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\ell^2} \right) \tag{2.17}$$

hyperparameters, $\boldsymbol{\theta} = \{\ell, \sigma_l, \sigma_n\}$, of the GP model can be obtained by optimizing the marginal log-likelihood. Assuming that tactile sensors can provide a normal vector to the contact point, one can integrate the surface normal in the GP model and in the covariance function. This approach exploits derivative observation both in training and inference. While it can better inform our model of the surface curvature, especially in the case of missing data, we observed no significant improvement, despite the computational sacrifice, compared to training with off-surface points. Besides, measuring surface normal vectors as derivative observation is more noisy and inaccurate than measuring off-surface points.

3 Efficient Configuration Exploration for Inverse Dynamics Acquisition

The work presented in this chapter has been published in *Khadivar, F., Gupta, S., Amanhoud, W. and Billard, A. "Efficient Configuration Exploration in Inverse Dynamics Acquisition of Robotic Manipulators." IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 1934-1941*

3.1 Introduction

The inverse dynamics (see Section 2.3) of a robotic manipulator is instrumental in precise robot control and manipulation. However, acquiring such a model is challenging, not only due to unmodeled non-linearities such as joint friction, but also from a machine learning perspective (e.g., input space dimension, amount of data needed). The accuracy of such models, regardless of the learning techniques, relies on proper excitation and exploration of the robot's configuration space, in order to collect a rich dataset.

In order to acquire an accurate ID model for a robot manipulator, in this chapter, we focus on generating sampling trajectories that efficiently explore the robot's joint space –that is, collecting data with maximum entropy. We propose a novel framework, Max-Information Configuration Exploration (MICE), that incrementally generates the richest data based on information theory (Taneja (1989)) and probabilistic approaches (Murphy (2012)). This framework can adopt various trajectory planners (e.g., sinusoids, quadratic, etc.) in that MICE applies to the parameter space of the trajectory generator. However, we offer a new path planning scheme by inducing stable limit cycles in the phase space of each robot joint. These limit cycles allow us to generate feasible trajectories in phase space (Khalil and Grizzle (2002)) and control the richness of the collected data not only in joint positions but also in joint velocities.

We benchmark our approach against conventional methods on a 7-DoFs robotic arm, KUKA LBR iiwa 14, by comparing the quality of the recorded data and the accuracy of the acquired models. To assess real-time robustness, we also study model performance in real robot

applications like trajectory tracking. Experimental evaluations reveal significant enhancement in configuration space exploration, model learning performance, and robotic evaluation.

In Section 3.3, we explain our exploration approach and compare the quality of collected data with those recorded from Fourier series. Then in Section 3.4, we validate that our approach can work with any model-learning method and architecture by using both Support Vector Regression (SVR) and Artificial Neural Network (ANN). We evaluate the performance of the models on a real robotic trajectory tracking task in Section 3.5.

3.2 Problem Statement

Some common sampling trajectories include the usage of Fourier series, as a family of sinusoids (Nguyen-Tuong et al. (2008c); Stürz et al. (2017); Swevers et al. (1997)):

$$q^i(t) = \sum_k A^i_{(k)} \sin\left(\omega^i_{(k)} t + \phi^i_{(k)}\right). \tag{3.1}$$

Despite being intuitive to implement, excitation trajectories parametrized with Fourier series often need to be sufficiently randomized to excite all frequencies of the system while remaining within the physical bounds (Vantilt et al. (2015)). Other approaches exploit online/offline optimization to obtain the optimal set of points (Du et al. (2014); Gautier and Khalil (1991); Presse and Gautier (1993)) from which are built the excitation trajectories. As the DoFs of the robotic manipulator increase, so does the computational complexity associated with finding the optimal points. This makes usage of such techniques to gather large quantity of data (e.g., more than 100,000 samples) for a high DoFs robotic manipulator practically infeasible.

We tackle the curse of dimensionality by optimally exploring in the parameters of our trajectory planner where the computation cost, as well as the frequency of the required optimizations, decrease drastically, simply because one point in parameter space represents a full trajectory in the joints' space.

More precisely at each increment, we select a vector of parameters that will result in trajectories with maximum average information.

3.3 Exploration Approach

3.3.1 Phase Space Path Planning

Capturing sufficient data complexity relies on visiting diverse pairs of joints' position and velocity $\{q, \dot{q}\}$. Thus instead of path planning solely in configuration space, we perform our path planning in the joints' phase space where $\{q, \dot{q}\}$ are the states of the desired trajectory. This enables us to build up data richness both in position and velocity through modulating path planning parameters. However, these trajectories have to satisfy feasibility constraints as

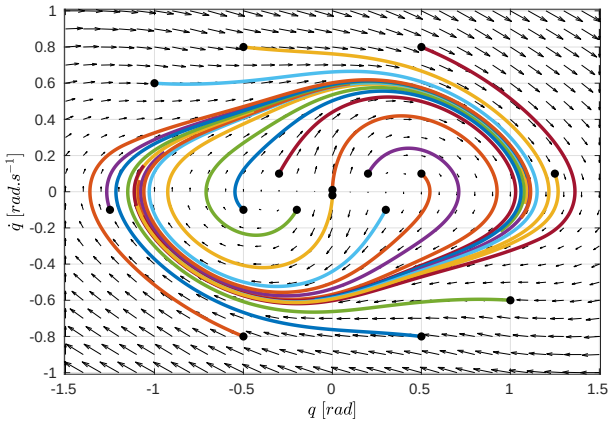


Figure 3.1: Various trajectories converging to a single stable limit cycle from different initial state $(q^i, \dot{q}^i)$, black dots. Arrows depict feasible motion direction in phase space generated by normalized states $(q^i/q^i_{max}, \dot{q}^i/\dot{q}^i_{max})$ as input to Eq. 3.2 and the DS in Eq. 3.4 with $\rho^i_c = 0.7$, $q^i_c = 0$, $\alpha^i = 1.0$ and $\beta^i = 0.25$.

not every trajectory is physically feasible in the phase space (Khalil and Grizzle (2002)).

To this end, we generate trajectories using a dynamical system (DS). The DS generates elliptical trajectories as limit cycles (see Section 2.2) in the phase space $(q, \dot{q})$ of each joint, while maintaining the feasibility constraints. For each joint, the phase space coordinates $(q^i \in \left[\underline{q}^i, \overline{q}^i\right], \dot{q}^i \in \left[\underline{\dot{q}}^i, \overline{\dot{q}}^i\right])$ are first converted into polar coordinates $(\rho^i \in \mathbb{R}^+, \theta^i \in [0, 2\pi])$:

$$\begin{cases} \rho^i = \sqrt{(q^i - q^i_c)^2 + \dot{q}^{i2}} \\ \theta^i = tan^{-1}\left(\dfrac{\dot{q}^i}{q^i}\right) \end{cases} \tag{3.2}$$

with $i \in [1, N]$ the joint index and $\underline{q}^i, \overline{q}^i, \underline{\dot{q}}^i, \overline{\dot{q}}^i \in \mathbb{R}$ the lower/upper bounds on the joint position and velocity. ρ^i is the polar radius, θ^i the phase angle, and q^i_c the center of the desired limit cycle on the q axis. The reverse transform can be achieved by:

$$\begin{cases} q^i = \rho^i\, cos\,(\theta^i) + q^i_c \\ \dot{q}^i = \rho^i\, sin\,(\theta^i). \end{cases} \tag{3.3}$$

The polar coordinates in phase space are provided as input to a DS specifying the desired polar dynamics to follow in the joint space:

$$\begin{cases} \dot{\rho}^i = -\alpha^i\,(\rho^i - \rho^i_d) \\ \dot{\theta}^i = -\beta^i\, e^{-\alpha^{i2}\,(\rho^i - \rho^i_d)^2} \end{cases} \tag{3.4}$$

where $\rho^i_d \in \mathbb{R}^+$ is the radius of the desired limit cycle while $\alpha^i \in \mathbb{R}^+$ and $\beta^i \in \mathbb{R}^+$ are tuneable parameters modulating the convergence speed. Note that as β^i is positive and q^i_c lies on the

q axis, meaning that limit cycles will have clockwise rotation and q-axis is perpendicular to the path. Therefore, all acquired trajectories from such a DS are always feasible for each joint. Figure 3.1 shows an examples of generated trajectories in a phase-space. From this figure, all trajectories, with different initial states, eventually converge to the desired limit cycle. With opting limit cycles of different radius and position, we can cover the entire phase space of the joint. As a result, a sequence of limit cycles can help us to stably and comprehensively explore a robot joints' phase space. Next, we explain how the succession of such DS is designed and employed in our exploration approach.

3.3.2 Maximizing Information

We refer to the data recorded during the robot's motion following the joint trajectory determined by the polar DS as the *swept data*. The continuous accumulation of the swept data between successive limit cycles builds up the dataset to be used for learning the inverse dynamics. For the k-th limit cycle let us define $\psi^i_{(k)} = \begin{bmatrix} \rho^i_{d(k)} \\ q^i_{c(k)} \end{bmatrix}$ as the parameters vector for i-th joint, i.e., $\psi^i_{(k)}$ gathers the desired center position (q^i_c) and radius (ρ^i_d) of the k-th limit cycle for joint i. Then for a N-joint robot, the parameters vector can be defined as: $\boldsymbol{\psi}_{(k)} = \begin{bmatrix} \psi^{1^T}_{(k)} & \cdots & \psi^{N^T}_{(k)} \end{bmatrix}^T$. Each $\boldsymbol{\psi}_{(k)}$ is a point in the *parameter space*, representing a unique combination of trajectory sets across all the joints where simple sequence of different $\boldsymbol{\psi}$ can generate a complex trajectory in the robot joints' phase space. Hence, instead of direct exploration in the joints' phase space, we will incrementally explore the trajectories' parameter space based on visited parameters:

$$\boldsymbol{\psi}_{(K+1)} = f(\boldsymbol{\Psi}) \tag{3.5}$$

where $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_{(k)}\}_{k=1,\dots,K}$ is the collection of parameters visited during the exploration with K being the total number increments. To determine the function $f(.)$ we take a probabilistic approach and select $\psi_{(K+1)}$ by maximizing the *information* entailed in the distribution of ψ over the parameter space:

$$\psi_{(K+1)} = \arg\max_{\boldsymbol{\psi}} \mathscr{I}(\boldsymbol{\psi}) \tag{3.6}$$

with $\mathscr{I}(\boldsymbol{\psi}) = -log(p(\boldsymbol{\psi}|\boldsymbol{\Psi}))$ and $p(\boldsymbol{\psi}|\boldsymbol{\Psi})$ the distribution of ψ over the parameter space given $\boldsymbol{\Psi}$. Therefore, the objective of the exploration is to select $\boldsymbol{\psi}_{(K+1)}$ so as to maximize the acquired information (Eq. 3.6). To this end, we first need to model $p(\boldsymbol{\psi}|\boldsymbol{\Psi})$. Here we use a kernel density estimator (KDE) (Murphy (2012)) as a non-parametric density model, allocating one cluster center per $\boldsymbol{\psi}_{(k)}$:

$$p(\boldsymbol{\psi}|\boldsymbol{\Psi}) = \frac{1}{K}\sum_{k=1}^{K} \mathscr{N}_{(k)}(\boldsymbol{\psi}; \boldsymbol{\psi}_{(k)}, \Sigma) \tag{3.7}$$

where $\Sigma = \text{diag}(\Sigma^i)$ for $i = 1, \ldots, N$ and each $\Sigma^i = \text{diag}(\sigma_{\rho_d^i}, \sigma_{q_c^i})$ is the smoothing parameter called *bandwidth*. Note that Σ determines the local effect of each cluster which can be modulated through $\sigma_{\rho_d^i}$ and $\sigma_{q_c^i}$ for radius and center respectively, which here are selected to be 5% of their valid range (e.i. $\sigma_{\rho_d^i} = .05(\rho^i_{max} - \rho^i_{min})$ and $\sigma_{q_c^i} = .05 q^i_{max}$). Using the estimated density (Eq. 3.7), we can now rewrite Eq. 3.6 and derive $\boldsymbol{\psi}_{(K+1)}$:

$$\boldsymbol{\psi}_{(K+1)} = \arg\min_{\boldsymbol{\psi}} \; log \sum_{k=1}^{K} \mathcal{N}_{(k)}(\boldsymbol{\psi}; \boldsymbol{\psi}_{(k)}, \Sigma). \tag{3.8}$$

Since the gradient of Eq. 3.7 has an analytical solution, Eq. 3.8 can be computed in real-time. Besides, we only need to use Eq. 3.8 at the end of each increment where all limit cycles are realized.

Technical note

Considering Eq. 3.8, as the main cost function, the gradient has the following form:

$$\frac{\partial f}{\partial \psi} = \Gamma \sum_{k=1}^{K} \mathcal{N}_{(k)}(\boldsymbol{\psi}; \boldsymbol{\psi}_{(k)}, \Sigma)(\boldsymbol{\psi} - \boldsymbol{\psi}_{(k)}) \tag{3.9}$$

where $\Gamma = \frac{\Sigma^{-1}}{\sum_{k=1}^{K} \mathcal{N}_{(k)}(\boldsymbol{\psi}; \boldsymbol{\psi}_{(k)}, \Sigma)}$. This gradient can be then used in fast optimization approaches like the gradient descent (Tseng and Yun (2009); Bierlaire (2015)). To avoid being trapped in local minimums, one can randomly select n initial guesses, and run the optimization over limited m iterations with convergence bound ϵ. Then the optimized variable is chosen as the optimum among the n acquired outputs. In this work and for the problem at hand, we selected $n = 100$, $m = 1000$ and $\epsilon = 0.001$.

3.3.3 Approach Summary and Evaluation

Our framework of data collection is summarized in Algorithm 1. Using MICE approach, Figure 3.2 depicts a simple example where through incrementally selecting $\boldsymbol{\psi}_{(k)}$ the distribution $p(\boldsymbol{\psi}|\boldsymbol{\Psi})$ becomes progressively more uniform, as the configuration space is covered more extensively.

To show the applicability of the approach, we have collected joint data and learned the inverse dynamics of a $N = 7$ robotic arm, where building up cross dependency across all joints is crucial. Figure 3.3a shows that distribution of all parameters $\boldsymbol{\psi}_{(k)}$ gradually converges to a uniform distribution due to constant decrease in KL-Divergence measure (Hershey and Olsen (2007)), with reference distribution being uniform, (as a dissimilarity measure to uniform distribution). With parameter distribution getting more uniform, maximizing the information will result in maximal entropy distribution (Taneja (1989)), see Figure 3.3a. Continuous increase of the entropy and KL-Divergence converging to zero confirms, as expected, that all the selected $\boldsymbol{\psi}$ are uniformly distributed across all the joints at the end of the data recording.

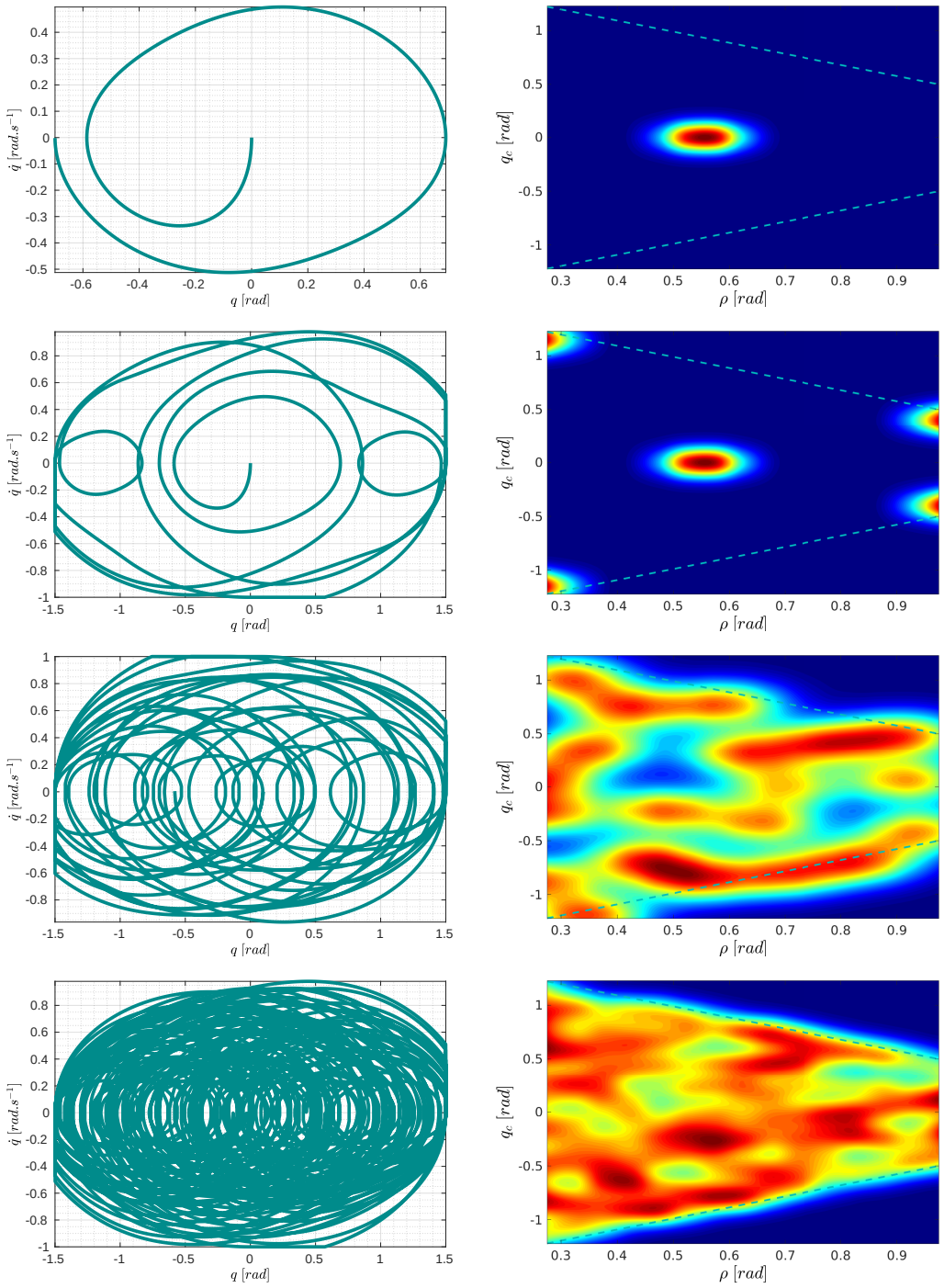
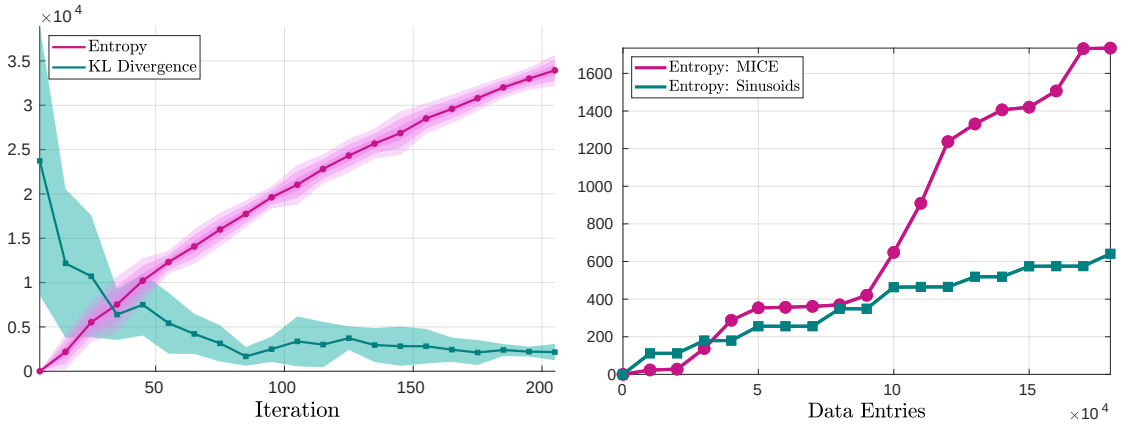


Figure 3.2: A One-joint example of MICE approach in parameter space ($\{\rho\,,\,q_c\}$) resulting in full configuration coverage. $K = 1, 5, 20$, and 75 is the increment number for rows from to bottom. Left side plots are generated trajectories as a succession of limit cycles. Right plots are the joint distribution for the selected parameters. Dash lines indicate the physical bounds over parameters from joint position and velocity limits.

Algorithm 1 Max-Information Configuration Exploration

Initialization:
$\boldsymbol{\psi}_0 \leftarrow$ random ▷ initial limit cycle parameters
Main loop:
for $t = 0 \rightarrow T$ **do**
  for $i = 1 \rightarrow N$ **do**
    if limit cycle Not completed **then**
      $[\theta, \rho]^i_t \leftarrow$ Eq. 3.2 $\leftarrow [q, \dot{q}]^i_t$
      $[\dot{\theta}, \dot{\rho}]^i_t \leftarrow$ Eq. 3.4 $\leftarrow [\rho^i_t, \psi^i_{(k)}]$
      $\theta^i_{t+1} = \theta^i_t + \dot{\theta}^i_t \Delta t$
      $\rho^i_{t+1} = \rho^i_t + \dot{\rho}^i_t \Delta t$
      $[q, \dot{q}]^i_{t+1} \leftarrow$ Eq. 3.3 $\leftarrow [\rho, \theta]^i_{t+1}$
      Data $\leftarrow$ Data $\cup\, [q, \dot{q}]^i_{t+1}$
      Check cycle completion $\leftarrow [\theta, \rho]^i_t$
    else
      $[q, \dot{q}]^i_{t+1} \leftarrow [q^i_t, 0]$ ▷ Wait for other joints
    end if
  end for
  if all limit cycles completed **then**
    $\boldsymbol{\Psi} \leftarrow \boldsymbol{\Psi} \cup \boldsymbol{\psi}_k$
    $p(\boldsymbol{\psi}|\boldsymbol{\Psi}) \leftarrow$ Eq. 3.7 ▷ Update distribution
    $\boldsymbol{\psi}_{k+1} \leftarrow$ Eq. 3.8 ▷ Update parameters
    $k \leftarrow k + 1$ ▷ New increment
  end if
end for

We compare MICE to Fourier series (Sinusoids) sampling method, in terms of swept data quality and learned model accuracy. Same as MICE, parameters (frequencies, amplitudes, and phase offsets) of the Fourier series are updated/randomized after a full period of the respective Sinusoid (Eq 3.1). Figure 3.3 reveals that the joint data recorded by MICE are distributed with substantially larger entropy than the data collected by sinusoids, proving more efficient spread of data over joint space. From Figure 3.3 we can also see that after some data entries, the entropy starts to grow faster when exploiting MICE, as it is better optimized to find information-rich trajectories. Next, we study and assess how swept data by MICE enhance the learning performance and model accuracy.



(a) MICE performance

(b) Data collection and entropy evolution

Figure 3.3: Algorithm 1 applied in parameters space of a 7-joint robotic arm, KUKA LBR iiwa 14: (a) Performance evaluation of MICE through entropy estimation and KL-Divergence measurement where reference distribution is uniform. The line indicates the mean, the shaded area shows the variance of 10 replicates. (b) Estimated entropy of data collected via sinusoids and MICE approach. Both (a) and (b) represent the same timespan, meaning that for approximately collecting 2×10^5 samples, 200 limit cycles are sequentially computed for each joint.

3.4 Model Learning

3.4.1 RBD-Based Model

For learning the inverse dynamics, one technique is to use the RBD model for predicting the target variable, i.e., the joint torque values:

$$\boldsymbol{\tau}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{\Phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \boldsymbol{\beta} \tag{3.10}$$

with β being the physical parameters of the robot. A robot manipulator has 10 physical parameters per link:

$$\boldsymbol{\beta}^i = [m\ mr_x\ mr_y\ mr_z\ I_{xx}\ I_{xy}\ I_{xz}\ I_{yy}\ I_{yz}\ I_{zz}]_i^T \tag{3.11}$$

where m is the link mass, (r_x, r_y, r_z) are the Cartesian coordinates of the link's centre of mass, while (I_{xx}) through (I_{zz}) are the inertia tensor components. Cumulatively, the physical parameters of the robot are $\boldsymbol{\beta} = [\boldsymbol{\beta}^1\ \boldsymbol{\beta}^2 \ldots \boldsymbol{\beta}^N]^T$.

Identification of the physical parameters by Parametric Least Squared Regression was carried out using the SciPy library (Virtanen et al. (2020)). On a 10-fold test set, the normalized error on predicted torque vector, τ, is presented in Table 3.1 for datasets acquired from MICE and Sinusoids. Throughout the chapter, errors are normalized w.r.t the magnitude of torque range for each joint. Table 3.1 confirms that the RBD model accuracy is improved by using MICE

Table 3.1: The joint-wise normalized mean squared errors in torque prediction by RBD models learned from two datasets.

Dataset	Joint-wise Prediction Error (%)						
	1	2	3	4	5	6	7
MICE	5.84	5.22	3.68	7.35	3.60	2.65	4.93
Sinusoids	5.84	5.34	3.85	7.51	3.65	2.72	4.93

dataset compared to the Sinusoids data. However, for both MICE and Sinusoids, the RBD model is unable to capture sufficient model complexities. The optimized physical parameters are later used to compute the non-linear error terms:

$$\epsilon^i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \tau^i - \boldsymbol{\Phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{\beta}^* \tag{3.12}$$

where i is the joint index and $\boldsymbol{\beta}^*$ are the optimized physical parameters representing the parametrically learned model.

3.4.2 Full and Error Models

Starting from Eq. 2.12, we pursue learning models that do not require explicitly identifying physical parameters, thereby offering more flexibility on capturing nonlinearities. We aspire to evaluate the quality of our swept data in training two mappings $\boldsymbol{f}_F(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ and $\boldsymbol{f}_E(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$. First the *Full* model, $\boldsymbol{\tau}_{full} = \boldsymbol{f}_F(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, and second the *Error* model, $\boldsymbol{\tau}_{err} = \boldsymbol{\Phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{\beta} + \boldsymbol{f}_E(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, which contains higher complexity. We use two learning techniques for training inverse dynamics models:

ν-Support Vector Regression

The framework for ν-SVR (Schiilkop et al. (1995); Vapnik (2013)) has been employed using the ThunderSVM library (Wen et al. (2018)). The key hyperparameters associated with model learning through ν-SVR are: (a) $\boldsymbol{\nu}$: regulator of the model complexity, (b) $\boldsymbol{\gamma}$: measure of the influence radius of support vectors, and (c) **C**: regularization parameter.

where here $\boldsymbol{\nu} = 0.1$ to limit model complexity and computation cost. These hyperparameters are then selected through cross validation and grid search. For the learned models the number of support vectors vary from 15000 to 50000. The training and testing performances of the Full models are presented in Table 3.2 where using MICE dataset guarantees higher precision than using the dataset of Sinusoids.

Artificial Neural Networks

The framework for the ANN (Bishop (1995)) has been implemented using the PyTorch library (Paszke et al. (2017)). Contrary to SVR models where one model can only predict a

Table 3.2: Normalized errors of torque prediction from Full models learned using ν-SVR with tuned hyperparameters. Both MICE and Sinusoids data have 150,000 sample points.

Joint	Training Error (%)		Testing Error (%)	
	MICE	Sinusoid	MICE	Sinusoid
1	2.15 ± 0.19	9.86 ± 0.21	2.96 ± 0.65	12.80 ± 0.18
2	0.32 ± 0.00	1.48 ± 0.00	9.87 ± 0.01	12.27 ± 0.01
3	0.23 ± 0.05	1.22 ± 0.03	2.88 ± 0.02	15.38 ± 0.22
4	0.41 ± 0.04	1.45 ± 0.00	4.34 ± 0.01	23.34 ± 0.02
5	0.56 ± 0.19	4.24 ± 0.08	0.88 ± 0.07	4.21 ± 0.20
6	0.54 ± 0.01	0.83 ± 0.01	1.35 ± 0.03	2.91 ± 0.27
7	0.18 ± 0.01	1.36 ± 0.03	0.32 ± 0.02	1.41 ± 0.06

Table 3.3: The torque vector prediction errors (normalized) and tuned hyperparameters for Full models learned by neural network. Both datasets have the same 150,000 sample points.

Dataset	Hyperparameters			Normalized Error (%)	
	α	L	n	Training	Testing
MICE	0.01	2	500	1.39 ± 0.62	1.53 ± 0.64
Sinusoids	0.05	3	25	14.17 ± 1.10	20.53 ± 0.44

single joint torque or the error term, a single neural network is capable of producing a multidimensional output. To learn the multidimensional joint torque and error prediction, the key hyperparameters associated with the shallow neural networks are: (a) Activation function, (b) Number of hidden layers L, (c) Neurons per each hidden layer n, and (d) Learning rate α. The activation function used in the networks is ReLU while the other hyperparameters were determined using grid search. The model accuracies for Full models, on their respective datasets, are provided in Table 3.3. Given Table 3.2 and Table 3.3, compared to Sinusoids, MICE provides more information-rich data, resulting in better learning performance regardless of the learning technique. Besides, models learned via ANN for both datasets promise higher precision than those learned by SVR. Therefore, we take ANN as the learning method for training Full and Error models on the comprehensive MICE data set, see Table 3.4.

Table 3.4: The normalized torque prediction error for Full and Error models. Bot models are learned by neural network via following tuned hyperparameters on MICE dataset with 2,200,000 sample points.

Model	Hyperparameters			Normalized Error (%)	
	α	L	n	Training	Testing
Full	0.01	2	500	0.62 ± 0.15	0.64 ± 0.16
Error	0.01	2	500	2.5 ± 0.086	2.62 ± 0.081

3.5 Model Evaluation

So far, we have learned and evaluated the acquired models on the training and testing sets receptively. Yet the high accuracy in the test set does not guarantee the same robustness and performance in real robotic tasks. Non-visited configurations and accumulation of prediction error are among the key sources of instability. Accordingly, we assess the trained models for both static and dynamic execution.

3.5.1 Static Test - Prediction of Gravity Compensation Torques

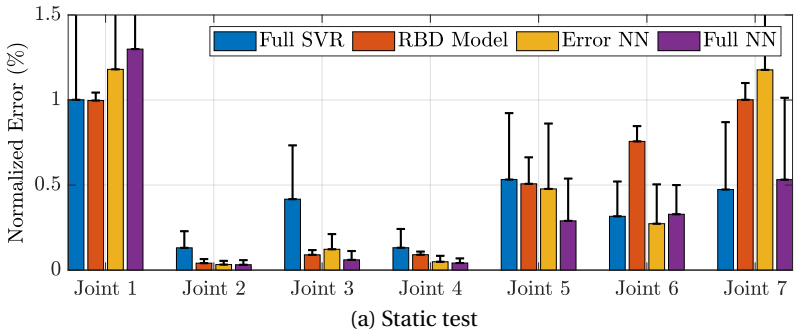
In plenty of robot controllers, the robot model is only used to compensate for static forces on the robot, referred to as gravity compensation torques. To test the efficacy of the learned models at predicting the gravity compensation torques, the manipulator was designated to reach randomly allocated stationary configurations in position control mode (a total of 200 configurations). The joint torque readings from the manipulator's sensors were used as the base line for comparing predictions from other models. For gravity compensation, the joint-wise error normalized over torque range are presented in Figure 3.4a. Given this, the models learned using Neural Networks have the most accurate predictions. Note that 2-end to 4-th joints are the most load bearing joints in KUKA LBR iiwa 14; therefore, the performance of gravity prediction is of higher importance there.

3.5.2 Dynamics Test - Trajectory Tracking Task

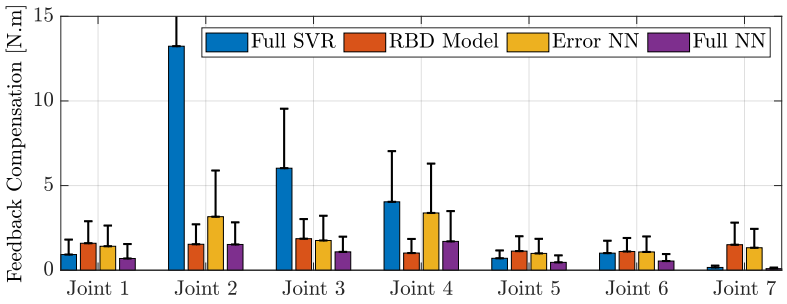
Robot dynamics can also be exploited to predict proper feedforward joint torques needed to achieve a desired trajectory. As an example of such a scenario, we designed a trajectory tracking task in the robot's joint space. In this task, the manipulator has to follow a sinusoidal test trajectory for a specified duration (here 180s).

The learned models are used to predict the feedforward torque based on the desired joint states $(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d)$. In conjunction to this, a proportional-derivative feedback (PID) controller (Ogata and Yang (2002)) is also implemented. For each model test, the PID gains are tuned such that the errors in following the desired joint-angles and joint-velocities remain in a specific bound (Craig (1989)). This is to ensure that the desired trajectory is perfectly track during each test.

Since the feedback torques regulate the tracking errors, low magnitude of these torques indicates the accuracy of feedforward torque predication, and less need for real-time compensation, Figures 3.4b. From amongst different models, the Neural Network based Full model required lower feedback compensation. This enhances robot stability, and allows more compliance as well as wider range of control gains.



(a) Static test



(b) Dynamic test

Figure 3.4: (a) Joint-wise normalized errors between the actual sensor torques and model predictions in gravity compensation mode. (b) Mean absolute values of feedback torques of a PID controller while exploiting learned models in a trajectory tracking task. The inverse dynamics is used as feedforward, and feedback torques are to regulate tracking errors. In both, the caps represent the variance in the normalized error.

3.6 Discussion and Summary

We introduced a novel framework, MICE, to explore robot joints' phase space and record rich data for model learning purposes. Our algorithm incrementally maximizes the information of swept data by generating stable limit cycles in joints' phase space. Studying the distribution of two datasets recorded by MICE and Sinusoids reveals that the former has significantly higher entropy and is distributed more uniformly. Subsequently, this has resulted in better model learning performance than the latter, regardless of the learning method. Moreover, to ensure the robustness of the acquired models, we have tested them on real robotic scenarios, such as evaluating the quality of gravity compensation and performance in trajectory tracking. The results show that the MICE dataset's Full model obtained by ANN guarantees the best accuracy and robustness among other learned models.

This work can further be studied and developed to improve the learned inverse dynamics. For instance, MICE can be exploited in online learning frameworks to generate rich excitation trajectories. Using incremental model learning approaches (Nguyen-Tuong and Peters (2010)) in conjunction with MICE is another potential topic of study which can improve the efficiency of model learning by only focusing on a desired workspace.

4 Self-Correcting Quadratic Programming-Based Control

Chapter 3 introduced methods to acquire an accurate model of a robot's inverse dynamics. However, due to model imperfections or structural changes, the model at hand and the reality do not match in many real-world scenarios, even if a precise model is previously learned. For instance, consider retraining the robot's model in a new gripper is mounted on the end-effector. In this case, retraining the model from scratch would be inefficient and costly. On the other hand, most model-based controllers' efficiency relies heavily on the model precision assumption: the model adequately captures the underlying robot/environment model. In this chapter, we take one model-based controller as an example and propose a control pipeline to relax this relatively strong assumption.

Quadratic Programming (QP)-based controllers allow many robotic systems, such as humanoids, to successfully undertake complex motions and interactions. However, these approaches rely heavily on adequately capturing the underlying model of the environment and the robot's dynamics. This assumption, nevertheless, is rarely satisfied, and we usually turn to well-tuned end-effector PD controllers to compensate for model mismatches.

Our approach builds upon existing inverse dynamics (ID) model learning techniques to improve the QP dynamics model over time (Nguyen-Tuong and Peters (2011); Chebotar et al. (2019); Chatzilygeroudis and Mouret (2018); Deisenroth et al. (2013)) and existing adaptive control methods (Ioannou and Sun (2012)) that can regulate the feedback term and change the cost function of the QP on-the-fly. This is different from previous methods of learning for QP-based controllers that primarily attempt to alter the QP cost function only (e.g., (Spitz et al. (2017); Lober et al. (2016))). Our main contributions are:

(i) An episodic procedure of learning residual ID model for QP-based control where we use an expressive and differentiable Gaussian Process with Rigid Body Dynamic (RBD) model as prior. In our method, ID is differentiable with respect to the optimization variables; hence, it actively exploits the learned ID model within the optimization of the QP-based control.

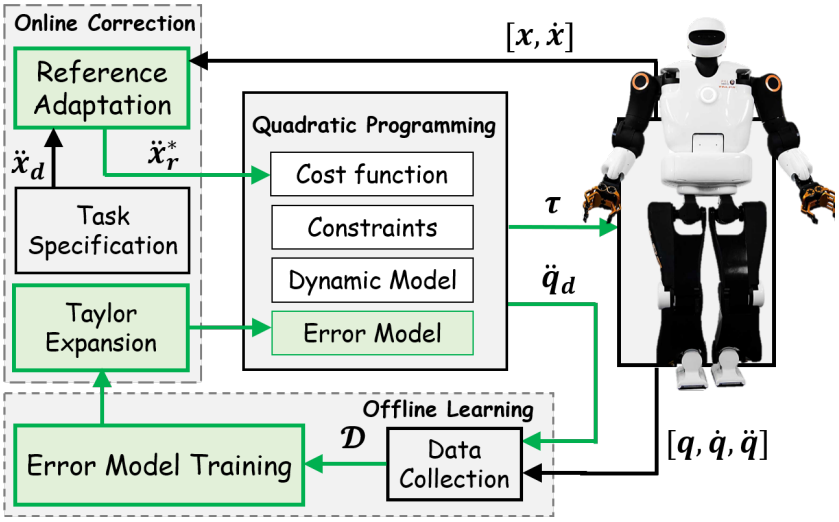


Figure 4.1: Our proposed framework for robot torque-control using a quadratic programming scheme. Green components, indicating our contributions, are developed in this study to compute the joint torques $\boldsymbol{\tau}$ for realizing a desired end-effector acceleration $\ddot{\boldsymbol{x}}_d$. Through online reference adaptation , given the feedback from robot end-effectors, and error model exploitation, using Taylor expansion, our approach enables the QP to correct itself such that the *expected* joint accelerations, $\ddot{\boldsymbol{q}}_d$, converge to the *actual* values, $\ddot{\boldsymbol{q}}$.

(ii) A novel scheme for continuous online reference adaptation in the cost function QP. To this end, we employ an adaptive controller that avoids manual tuning, addresses model uncertainties online, and results in a faster convergence for ID model learning.

We extensively validate our approach in simulations and perform experiments in a physical robotic setup. Our results showcase that our method (i) is able to compensate for large model inaccuracies in little interaction time (i.e., in a few trials), and (ii) consistently outperforms the baselines.

At the time of writing, the work presented in this chapter is under the second round of review as *Khadivar, F., Chatzilygeroudis, K., and Billard, A. "Self-Correcting Quadratic Programming-Based Robot Control." IEEE Transactions on Systems, Man and Cybernetics: Systems, 2022.* Konstantinos Chatzilygeroudis has equally contributed to the mentioned paper. More specifically, he is the main contributor to the implementation of GP learning and the foundation of simulated experiments. In this chapter, the theoretical work, simulated and physical robot experiments conducted by the thesis author are exclusively reported.

4.1 Introduction

In multi-body robotic systems, we need to carefully coordinate a large number of degrees of freedom and interaction forces. Such coordination becomes more challenging when operating in taskspace, even for the simplest tasks, e.g., a bimanual robot interacting with an object (Chen

et al. (2017)). Quadratic programming (QP)-based methods provide a principled control framework to tackle these challenges (Escande et al. (2014); Berenson et al. (2011); Bouyarmane and Kheddar (2017)). In QP, task requirements are formulated as an optimization process for minimizing an objective function together with satisfying constraints such as joint limits, system dynamics, and contact forces (Zhang et al. (2004); Baerlocher and Boulic (2004); Collette et al. (2007)). Due to smart formulation and modern computation, this optimization process can run up to 1 KHz, allowing for high-frequency control even in humanoids (Escande et al. (2014); Righetti and Schaal (2012); Bouyarmane and Kheddar (2011)).

Control structures based on optimization are used widely in general-purpose simulations and automatic motion synthesis. Among early studies of interest, Abe et al. (2007) controlled the balancing of animated humans by integrating constraints, namely the frictional and non-planar contact model. Later, De Lasa et al. (2010) and Coros et al. (2010) extended the approach and incorporated physical features into prioritized levels of optimization for locomotion with periodic contact switching. QP is an example of an optimization-based controller and has also been employed frequently in various applications in robotics such as trajectory planning/tracking (Li et al. (2016b)), multi-body control (Zhang et al. (2004)), multi-task planning (Salini et al. (2011)), and dynamic balancing (Li et al. (2016c)).

However, similar to most model-based controllers (Nakanishi et al. (2008)), the key assumption of any QP-based control is the precision of the model and whether it adequately captures the underlying robot/environment model. This is, of course, rarely the case since in many real-world scenarios, the model at hand and the real model do not match. Even when the system is coupled with accurate state estimators and high-gain PID feedback, real-world experiments are subject to frequent failures due to model imperfections, friction, actuator nonlinearities, etc. (Nguyen-Tuong and Peters (2011)). In practice, for these reasons, configuring a QP-based controller for a convoluted robot (e.g., humanoid) or complex interactions (e.g., handling objects) almost always involves a great deal of hand-tuning of the model parameters and the cost function for each specific instance of the task (Modugno et al. (2016)). In this study, we aim to address these limitations of QP-based controllers. We focus on the problem of model imprecision and feedback inadequacy of the control law.

Humans and animals have a remarkable way of performing new tasks or adapting to unforeseen situations. They learn from their mistakes and, by trial-and-error, master new skills. We envision a similar robotic system that learns through trial-and-error: it tries to achieve the task with a QP-based controller, fails (e.g., the box slips from the hands of the robot), and tries again until it manages to realize the goal. This adaptation is functional only if it is fast; imagine, for example, having to wait 2 days for a robot to learn how to perform a search and rescue scenario task. Thus we would like the procedure to happen in as short an interaction time as possible[1] (Chatzilygeroudis et al. (2019)).

[1] Minimizing the interaction time with the system is equivalent to minimizing the number of trials in episodic settings

Therefore, the main question that arises here is: how can we improve a QP-based controller as per previous trials? Assuming an accurate/perfect QP solver, three central elements can be updated: (i) the *task specification* (i.e., end-effector desired accelerations), (ii) the *cost function* of the QP, and (iii) the *model* of the QP. The first would require an external oracle to give us feedback on whether or not we were performing the task well, and then one would have to find which is the best oracle to do so. We therefore assume that the task specifications are well-designed and we do not update them. Altering the cost function freely is likewise dangerous. The QP solver might fail or produce weird motions, and this is why most related approaches model the cost function as a series of waypoints or attractors/repulsors (Spitz et al. (2017); Lober et al. (2016)). In contrast, improving the model of the QP with data gathered from the physical trials will make the task of the QP solver easier and more accurate.

In this chapter, we focus on QP-based ID controllers and investigate means by which we can update the *cost function* to ensure stability of the system, and improve the precision of the *ID model* efficiently. More precisely, we formulate an ID model learning procedure to improve the model of the QP and show that it applies to a variety of robots with different dynamics. We then introduce a novel QP-based control scheme that is able to overcome model inaccuracies and large model mismatches by combining the slow ID model learning with a fast online adaptive control law in taskspace to regulate the cost function of the QP(see Figure 4.1). Using adaptive control per se for robot control is not new, however, to the best of our knowledge, we have for the first time employed online adaptation of nonlinearities and model uncertainties to close the control loop for QP-based controllers.

4.2 Quadratic Programming-Based Control

For many applications in robotics, it is more convenient and intuitive to design the desired behavior in the Cartesian coordinate system of the end-effector(s), also known as the taskspace. In the ID formulation, we have as input desired end-effectors accelerations $\ddot{\boldsymbol{x}}_d \in \mathbb{R}^l$ (l is the dimension of the taskspace), and we would like to find the joint-level torques needed to achieve these accelerations. The equations of motion and constraint equations for a robot can be described as (Feng et al. (2014)):

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \boldsymbol{S\tau} + \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W} \\ \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} &= \ddot{\boldsymbol{x}}_r \end{aligned} \tag{4.1}$$

where $\boldsymbol{q} \in \mathbb{R}^j$ is the full state of the system (with j being the number of DoFs of the robot, including the 6-DoFs of the floating base, if present)[2], $\boldsymbol{x} \in \mathbb{R}^l$ is the concatenation of the poses (containing position and orientation) in Cartesian space of all the contact points (if present), $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{j\times j}$ is the inertia matrix, $\boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^j$ is the sum of the gravitational, centrifugal and Coriolis forces, $\boldsymbol{S} \in \mathbb{R}^{j\times j}$ is a selection matrix where the first 6 rows are all zeros and the rest is the identity matrix, $\boldsymbol{W} \in \mathbb{R}^{6n_c\times j}$ is the concatenation of all n_c contact wrenches (in world

[2]We denote time derivatives with an upper dot: e.g., $\dot{x}$.

frame), $\boldsymbol{J} \in \mathbb{R}^{6n_c \times j}$ is the concatenation of the Jacobians of all the contact points, and $\boldsymbol{\tau} \in \mathbb{R}^j$ is the vector containing the control torques of all DoFs of the system. We can re-write the equations of motion as (Feng et al. (2014)):

$$\begin{bmatrix} \boldsymbol{M}(\boldsymbol{q}) & -\boldsymbol{S} & -\boldsymbol{J}(\boldsymbol{q})^T \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\tau} \\ \boldsymbol{W} \end{bmatrix} + \boldsymbol{C_g}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = 0. \tag{4.2}$$

This formulation is interesting as given a state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the equations of motion are linear with respect to $\begin{bmatrix} \ddot{\boldsymbol{q}} & \boldsymbol{\tau} & \boldsymbol{W} \end{bmatrix}^T$. By defining $\mathscr{X} = \begin{bmatrix} \ddot{\boldsymbol{q}} & \boldsymbol{\tau} & \boldsymbol{W} \end{bmatrix}^T$ in this chapter, we can now express the ID formulation as a QP-based whole-body control problem (Feng et al. (2014)):

$$\begin{aligned} \min_{\mathscr{X}} \; & -\frac{1}{2}\mathscr{X}^T \boldsymbol{G} \mathscr{X} + \boldsymbol{g}^T \mathscr{X} \\ \text{s.t. } & \boldsymbol{H}_E \mathscr{X} = \boldsymbol{b}_E \\ & \boldsymbol{H}_I \mathscr{X} \geq \boldsymbol{b}_I \end{aligned} \tag{4.3}$$

where we are optimizing tasks of the form $\frac{1}{2}\|\boldsymbol{H}\mathscr{X} - \boldsymbol{b}\|^2$, and where $\boldsymbol{G} = \boldsymbol{H}^T\boldsymbol{H}$ and $\boldsymbol{g} = -\boldsymbol{H}^T\boldsymbol{b}$. In particular, we turn the equations of motion into equality constraints ($\boldsymbol{H}_E$ and $\boldsymbol{b}_E$), and we turn joint limits and other constraints, such as friction cone or center of pressure constraints, into inequality constraints ($\boldsymbol{H}_I$ and $\boldsymbol{b}_I$). We then define desired accelerations of some end-effector by filling $\boldsymbol{G}$ and $\boldsymbol{g}$ appropriately[3] (the QP task objectives). To make things more concrete, imagine a manipulator which is rigidly attached to the world, and we treat the base of its gripper as the end-effector. In this case, $\boldsymbol{x}$ represents the position and orientation of the base of the gripper of the manipulator (see Section 4.4.1 for an example on how to fill $\boldsymbol{G}$ and $\boldsymbol{g}$). When we have multiple tasks with different weights w_i, we can decompose $\boldsymbol{H}$ and $\boldsymbol{b}$ as $\boldsymbol{H} = \begin{bmatrix} \mathrm{w}_1\boldsymbol{H}_1^T, \mathrm{w}_2\boldsymbol{H}_2^T, \ldots, \mathrm{w}_n\boldsymbol{H}_n^T \end{bmatrix}^T$, and $\boldsymbol{b} = \begin{bmatrix} \mathrm{w}_1\boldsymbol{b}_1, \mathrm{w}_2\boldsymbol{b}_2, \ldots, \mathrm{w}_n\boldsymbol{b}_n \end{bmatrix}^T$.

In this study, we compute the reference end-effector accelerations $\ddot{\boldsymbol{x}}_r$ by:

$$\ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}_f + \ddot{\boldsymbol{x}}_d \tag{4.4}$$

where $\ddot{\boldsymbol{x}}_d$, the end-effector desired acceleration, is the feed forward control term specified by a higher-level controller and can change over time based on the defined current task. $\ddot{\boldsymbol{x}}_f$ is the feedback term which closes the control loop. Previous work has usually applied PID or PD control structure to close the loop: $\ddot{\boldsymbol{x}}_f = -k_p(\boldsymbol{x} - \boldsymbol{x}_d) - k_v(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_d)$. Finding proper gains ($k_p \in \mathbb{R}^+$ and $k_v \in \mathbb{R}^+$) for each task requires heavy gain tuning with no stability guarantee, and no control over the transient error behavior. In this chapter, we close the control loop via an MRAC control scheme. In other words, $\ddot{\boldsymbol{x}}_r$ is derived from a nonlinear adaptive controller presented in Section 4.3.1. By doing so, we modulate $\ddot{\boldsymbol{x}}_r$ in an online fashion to follow a desired dynamical system, thereby controlling the transient behavior of the error signal. Moreover, we guarantee the stability of the system in the feedback line with appropriate adaptive laws.

[3] We use the `whc` library: https://github.com/costashatz/whc.

4.3 Approach

We propose a novel QP-based control scheme, called the Self-Correcting QP-based Control framework (SCQP, see Figure 4.1), where a nonlinear adaptive controller computes the reference accelerations for the QP-based ID (Section 4.3.1), and a learning procedure improves the ID model of the QP (Section 4.3.2). Overall, in the SCQP we adopt an episodic learning scheme and perform the following steps (see also Algorithm 2):

1. Design the task specifications: $\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t), \ddot{\boldsymbol{x}}_d(t)$.
2. Configure the adaptive controller and model learning procedure.
3. Perform an episode. For each time-step:
   (a) Compute the reference accelerations, $\ddot{\boldsymbol{x}}_r$ using our nonlinear adaptive controller (Section 4.3.1).
   (b) Compute the cost function for the QP given $\ddot{\boldsymbol{x}}_r$;
   (c) Get the torques $\boldsymbol{\tau}$ from the QP with the updated cost function, and the learned ID model, $\boldsymbol{h}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ via linearization.
   (d) Apply the torques to the robot and collect data.
   (e) Update the adaptive controller at each time-step.
4. Learn the ID model with Gaussian Processes with all the collected data.
5. Go back to step 3 until convergence.

Throughout this chapter, we use n, l, and $m \in \mathbb{N}$ to refer to the dimension of the state-space, taskspace, and control input, respectively. For easier reference, in Table 4.1, we list all the notations necessary to follow our approach and derivations, hereafter refer to Table 4.1 for the variable-related dimensions. We use subscripts r and d for indexing *reference* and *desired* variables and utilize the following conventions throughout the article: typeface for scalars (e.g., a), lowercase bold font to represent vectors (e.g., $\boldsymbol{a}$), and uppercase bold font to refer to matrices (e.g., $\mathbf{A}$). For brevity, we drop the variable-related indexing of each variable in Table 4.1.

4.3.1 Taskspace Adaptive Control

Let us define $\boldsymbol{\zeta} = [\boldsymbol{x}^T, \dot{\boldsymbol{x}}^T]^T$ to be the states of the end-effector in the taskspace. Then, the objective of our adaptive controller is to ensure that the $\boldsymbol{\zeta}$ converge to the desired states $\boldsymbol{\zeta}_d = [\boldsymbol{x}_d^T, \dot{\boldsymbol{x}}_d^T]^T$.

Table 4.1: Core notations

Symbol	Dimension	Description
$\boldsymbol{x}$	$\mathbb{R}^l$	Robot end-effector Cartesian position
$\boldsymbol{\zeta}$	$\mathbb{R}^n$	Robot taskspace states
$\boldsymbol{\nu}$	$\mathbb{R}^m$	Adaptive output control signal
$\boldsymbol{\Phi}(\cdot)$	$\mathbb{R}^p$	Vector of basis functions
$\mathbf{P}$	$\mathbb{R}^{n\times n}$	A symmetric positive definite matrix
$\mathbf{Q}$	$\mathbb{R}^{n\times n}$	A positive definite matrix
$\mathbf{A}$	$\mathbb{R}^{n\times n}$	State matrix of a dynamical system
$\mathbf{B}$	$\mathbb{R}^{n\times m}$	Input matrix of a dynamical system
$\boldsymbol{r}(t)$	$\mathbb{R}^n$	Input signal of the reference model
$\bar{\Psi}_\upsilon$	$\mathbb{R}^{m\times n_\upsilon}$	Estimated control gain for $\upsilon = \boldsymbol{\zeta}, \boldsymbol{r}, \boldsymbol{\phi}$
Λ_υ	$\mathbb{R}^{m\times n_\upsilon}$	Adaptation gain for $\upsilon = \boldsymbol{\zeta}, \boldsymbol{r}, \boldsymbol{\phi}$

Reference Model

The states selection depends on the task requirement; for instance, one might define $\boldsymbol{\zeta} = \dot{\boldsymbol{x}}$ where $\dot{\boldsymbol{\zeta}}_d$ can be directly given or derived from a stable dynamical system $\dot{\boldsymbol{\zeta}}_d = \boldsymbol{f}_d(\boldsymbol{\zeta})$. We consider a general case where the full states of the robot end-effector, $\boldsymbol{\zeta}$, need to follow a desired reference model given by:

$$\dot{\boldsymbol{\zeta}}_r = \mathbf{A}_r\boldsymbol{\zeta}_r + \mathbf{B}_r\boldsymbol{r}(t) \tag{4.5}$$

where $\boldsymbol{r}(t)$ is a bounded regulation signal. $\mathbf{A}_r$, $\mathbf{B}_r$, and $\boldsymbol{r}(t)$ are design parameters to shape the reference model and have to be such that the dynamic model (4.5) is stable, from a Lyapunov perspective, at $\boldsymbol{\zeta}_d$, meaning that $\|\boldsymbol{\zeta}_r - \boldsymbol{\zeta}_d\| \to 0$ as $t \to \infty$. In addition to stable control, this model allows modulating the *transient behavior* of the stable convergence. For instance, matrix $\mathbf{A}_r$ controls how fast $\boldsymbol{\zeta}_r$ converges to $\boldsymbol{\zeta}_d$ while $\mathbf{B}_r$ regulates $\boldsymbol{\zeta}_r$ to track $\boldsymbol{\zeta}_d$. The regulation signal, $\boldsymbol{r}(t)$, as the reference signal could be computed from $\boldsymbol{r}(t) = -\mathbf{B}_r^\dagger\mathbf{A}_r\boldsymbol{\zeta}_d$ with $\mathbf{B}_r^\dagger$ being the pseudo inverse of $\mathbf{B}_r$.

End-Effector Model

Given $\boldsymbol{\zeta}$ being the concatenation of end-effector position, $\boldsymbol{x}$, and velocity, $\dot{\boldsymbol{x}}$, we can formulate a nonlinear dynamic model that governs the derivative of these states, $\dot{\boldsymbol{\zeta}}$ by having the acceleration, $\ddot{\boldsymbol{x}}_f$, from Eq. 4.4, to be the input:

$$\dot{\boldsymbol{\zeta}} = \mathbf{A}\boldsymbol{\zeta} + \mathbf{B}\boldsymbol{\nu} + \mathrm{F}(\boldsymbol{\zeta}_d, \boldsymbol{\zeta}) \tag{4.6}$$

and $\boldsymbol{\nu} = \ddot{\boldsymbol{x}}_f$ is the control effort. Matrices $\mathbf{A}$, and $\mathbf{B}$ are unknown and to be determined by adaptation laws. $\mathrm{F}(\cdot) \in \mathbb{R}^n$, either fully or partially unknown, is a smooth function, and we approximate it by employing universal function approximators $\mathrm{F}(\cdot) = \Psi_\phi^*\boldsymbol{\Phi}(\cdot)$ explained

in Appendix A.2. Such a model implies that to perfectly track the desired states $\boldsymbol{\zeta}_d$, the controller effort has to take into account unmodeled nonlinearities and adapt for unexpected uncertainties.

In a nutshell, the dynamic model (4.5) and the defined $\boldsymbol{r}(t)$ build the reference model for MRAC that is utilized to find $\ddot{\boldsymbol{x}}_f$ for Eq. 4.4.

Control Rules and Adaptive Laws

To ensure that $\boldsymbol{\zeta}$ track the reference dynamics model (4.5) in the presence of nonlinearities and uncertainties, we propose the following control rule for the system (4.6):

$$\boldsymbol{\nu} = \Psi_\zeta \boldsymbol{\zeta} + \Psi_r \boldsymbol{r}(t) + \Psi_\phi \boldsymbol{\Phi}(\boldsymbol{e}) \tag{4.7}$$

where Ψ_ζ, Ψ_r, and Ψ_ϕ are approximated online. Our taskspace adaptive controller takes $\boldsymbol{\zeta}$ and $\boldsymbol{\zeta}_d$ as input, and outputs $\ddot{\boldsymbol{x}}_r$ to be fed to QP optimization:

$$\ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}_d + \bar{\Psi}_\zeta \boldsymbol{\zeta} + \bar{\Psi}_r \boldsymbol{r}(t) + \bar{\Psi}_\phi^T \boldsymbol{\Phi}(\boldsymbol{e}) \tag{4.8}$$

in which $\bar{\Psi}_\zeta$, $\bar{\Psi}_r$, and $\bar{\Psi}_\phi$ are the approximated matrices in Eq. 4.7, based on the following adaptation laws for control parameters:

$$\begin{aligned} \dot{\bar{\Psi}}_\zeta &= -\Lambda_\zeta \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{\zeta}^T \\ \dot{\bar{\Psi}}_r &= -\Lambda_r \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{r}^T(t) \\ \dot{\bar{\Psi}}_\phi &= -\Lambda_\phi \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{\Phi}(\boldsymbol{e})^T \end{aligned} \tag{4.9}$$

where Λ_ζ, Λ_r, and Λ_ϕ are positive definite matrices that tune the convergence rate of the adaptive gains. $\mathbf{P}$ with $\mathbf{A}_r$ satisfy $\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$ as the necessary and sufficient stability condition for the reference model (4.5); see Appendix A.1 for the stability proof. Note that, in the first trial, we need to initialize the adaptive gains $\bar{\Psi}_\zeta$, $\bar{\Psi}_r$, and $\bar{\Psi}_\phi$ in Eq. 4.10 with imposed upper and lower bounds. The adaptation rate has to be faster than the actual dynamics. Once a task is completed, we can store the trained adaptive gains and use them as the initial guess for new trials. Due to the adaptation laws (4.10) the proposed controller is now able to cope online with errors in the QP dynamic model and unforeseen uncertainties.

As for the whole system stability when using QP-based controllers, Bouyarmane and Kheddar (2017) show that such a controller, under certain assumptions, is stable in terms of solution existence, uniqueness, robustness to perturbation, and continuity. Also from another perspective, a QP-based controller can be seen as a one-step horizon model predictive controller (MPC). Regarding the stability of MPC, in (Anderson and Moore (2007)) and (Limón et al. (2006)), authors prove (i) the recursive feasibility by showing the existence of a feasible control sequence at all time instants when starting from a feasible initial point and (ii) the stability by showing that the optimal cost function is a Lyapunov function.

4.3.2 Inverse Dynamics Learning Procedure

The task of ID is to provide a model $\boldsymbol{h}^*$ that gives us the torques needed to apply to the system to achieve some desired joint accelerations in a particular robot state:

$$\boldsymbol{\tau} = \boldsymbol{h}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d). \tag{4.10}$$

This is a classic reformulation of Eq. (4.1) in order to "see" the equation as a data-driven model/mapping to learn (see (Nguyen-Tuong and Peters (2011, 2010)) for a detailed overview of ID model learning). This is a supervised learning task, and we can employ any suitable learning algorithm. Learning ID model can give us accurate models that can operate inside a control loop and improve tracking performance (Nguyen-Tuong and Peters (2010)). To reduce the sample complexity, or in other words to reduce the number of samples needed for achieving good accuracy, we can learn the difference from an available analytic model $\bar{\boldsymbol{h}}$:

$$\boldsymbol{h}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) = \bar{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) + \boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) \tag{4.11}$$

where $\bar{\boldsymbol{h}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + \boldsymbol{C}_{\boldsymbol{g}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W}$. In essence, we insert prior information coming from our inaccurate yet useful analytical RBD modeling.

In order to be able to exploit the ID model inside the optimization process, it needs to be linear with respect to the optimization variables $\mathscr{X} = \begin{bmatrix} \ddot{\boldsymbol{q}}_d & \boldsymbol{\tau} & \boldsymbol{W} \end{bmatrix}^T$ of the QP. This is required to take full advantage of the ID model that otherwise operates as a *static offset*. Thus, the error model $\boldsymbol{e}_{\boldsymbol{h}}(.) \in \mathbb{R}^j$ must not violate this linearity constraint. At the same time, training a linear model for $\boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_{\mathrm{d}})$ would deteriorate the performance and reduce the flexibility of both the model and the controller.

To overcome this limitation, we propose using more expressive and differentiable models and linearizing them around the current state. In particular, if we assume that the system is in a state $(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t)$, we take the first two terms of the Taylor series expansion of $\boldsymbol{e}_{\boldsymbol{h}}(\cdot,\cdot,\cdot)$:

$$\boldsymbol{h}^*(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_{t+1}) = \bar{\boldsymbol{h}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t) + \boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t) + (\ddot{\boldsymbol{q}}_{t+1} - \ddot{\boldsymbol{q}}_t) \frac{\partial \boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})}{\partial \ddot{\boldsymbol{q}}} \Big|_{\substack{\boldsymbol{q}=\boldsymbol{q}_t \\ \dot{\boldsymbol{q}}=\dot{\boldsymbol{q}}_t \\ \ddot{\boldsymbol{q}}=\ddot{\boldsymbol{q}}_t}} \tag{4.12}$$

where $\ddot{\boldsymbol{q}}_{t+1}$ contains the desired joint accelerations and is one of the variables optimized by the QP-based controller (i.e., $\ddot{\boldsymbol{q}}_d \equiv \ddot{\boldsymbol{q}}_{t+1}$). Here, it is important to note that for QP optimization the $\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t$ take fixed values that cannot change during an optimization at each time-step.

This linearization yields a loss in expressivity but allows us to insert the models inside the QP-based controller. Nevertheless, because our QP-based controller runs at high frequency (usually $\geq$ 200Hz), the linearized version of the model captures quite well the behavior of the full model in the optimization range and, as we show in the experiments, does not affect the system performance.

Gaussian Processes for Inverse Dynamics Learning

We use Gaussian Process Regression (GP) (Rasmussen and Williams (2006)) to learn the ID model. We choose GPs because they are accurate, generalize well, and, hence, are suitable for learning from few data points (Deisenroth et al. (2013); Chatzilygeroudis et al. (2017)). Another key property of GPs, when combined with prior information, is that they are guaranteed to fall back, in regions far from the data, to the prior model. This property ensures that the QP optimization, which is sensitive to the model it accepts, never fails. In preliminary experiments with neural networks, we observed frequent failures in QP optimization; see also Section 4.3.2.

As inputs, we use tuples made of the state vector $\tilde{\boldsymbol{q}} = (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t)$. As training targets, we use the difference between the prediction of the analytic model and the actual command sent: $\boldsymbol{e}_t = \bar{\boldsymbol{h}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_{t+1}) - \boldsymbol{\tau}_t$, where $\boldsymbol{\tau}_t$ is the torque command sent at time t. We use independent GPs to model each dimension of the difference vector $\boldsymbol{e}_t$. For each dimension d of $\boldsymbol{e}_t$, the GP is computed as ($k_{\boldsymbol{e}_d}$ is the kernel function):

$$\hat{\boldsymbol{e}}_d(\tilde{\boldsymbol{q}}) \sim \mathcal{GP}(\mu_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}, k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{q}}'))). \tag{4.13}$$

Assuming $D^d_{1:N} = \{\boldsymbol{e}_d(\tilde{\boldsymbol{q}}_1), ..., \boldsymbol{e}_d(\tilde{\boldsymbol{q}}_N)\}$ is a set of observations, we can query the GP at a new input point $\tilde{\boldsymbol{q}}_*$:

$$p(\hat{\boldsymbol{e}}_d(\tilde{\boldsymbol{q}}_*)|D^d_{1:N}, \tilde{\boldsymbol{q}}_*) = \mathcal{N}(\mu_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*), \sigma^2_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*)). \tag{4.14}$$

The mean and variance predictions of this GP are computed using a kernel vector $\boldsymbol{k}_{\hat{\boldsymbol{e}}_d} = k(D^d_{1:N}, \tilde{\boldsymbol{q}}_*)$, and a kernel matrix $K_{\hat{\boldsymbol{e}}_d}$ with entries $K^{ij}_{\hat{\boldsymbol{e}}_d} = k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_j)$:

$$\begin{aligned} \mu_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*) &= \boldsymbol{k}^T_{\hat{\boldsymbol{e}}_d} K^{-1}_{\hat{\boldsymbol{e}}_d} D^d_{1:N} \\ \sigma^2_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*) &= k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*, \tilde{\boldsymbol{q}}_*) - \boldsymbol{k}^T_{\hat{\boldsymbol{e}}_d} K^{-1}_{\hat{\boldsymbol{e}}_d} \boldsymbol{k}_{\hat{\boldsymbol{e}}_d}. \end{aligned} \tag{4.15}$$

We use the exponential kernel (Rasmussen and Williams (2006)) in this study:

$$k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_p, \tilde{\boldsymbol{q}}_q) = \sigma^2_d \exp(-\frac{1}{2}(\tilde{\boldsymbol{q}}_p - \tilde{\boldsymbol{q}}_q)^T \boldsymbol{\Lambda}^{-1}_d (\tilde{\boldsymbol{q}}_p - \tilde{\boldsymbol{q}}_q)) + \delta_{pq}\sigma^2_{n_d} \tag{4.16}$$

where δ_{pq} equals 1 when $p = q$ and 0 otherwise, and $[\boldsymbol{\Lambda}_d, \sigma^2_d, \sigma^2_{n_d}]$ is the vector of hyper-parameters of the kernel (length scales for each dimension of the input, signal variance and noise).

To linearize the learned model around a query point, we need to compute the derivative of our GPs. The derivative is another GP, and its existence depends on the differentiability of its kernel function (Rasmussen and Williams (2006)). In our particular case, the squared exponential kernel that we use is infinitely differentiable, and the associated GP has infinitely many derivatives. In this study we do not use the predicted GP variance, and we only detail the derivatives of $\mu_{\hat{\boldsymbol{e}}_d}$ with respect to the input point $\tilde{\boldsymbol{q}}_*$. If we look at Eq. 4.15 closely, only $\boldsymbol{k}_{\hat{\boldsymbol{e}}_d}$ depends on the input point $\tilde{\boldsymbol{q}}_*$, meaning that we just need to differentiate the kernel function.

Assuming that we have only one sample for training $\tilde{\boldsymbol{q}}_i$, we can compute the derivative of the kernel as follows:

$$\begin{aligned}\frac{\partial k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_*)}{\partial \tilde{\boldsymbol{q}}_*} &= \frac{\partial\left(\sigma_d^2 \exp(-\frac{1}{2}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i)^T \boldsymbol{\Lambda}_d^{-1}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i))\right)}{\partial \tilde{\boldsymbol{q}}_*} \\ &= \frac{\partial\left(-\frac{1}{2}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i)^T \boldsymbol{\Lambda}_d^{-1}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i)\right)}{\partial \tilde{\boldsymbol{q}}_*} k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_*) \\ &= -\boldsymbol{\Lambda}_d^{-1}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i) k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_*).\end{aligned} \tag{4.17}$$

It is trivial to generalize/compute the gradient when considering a set of training points.

Algorithm 2 Self-Correcting QP-based Control

```
Design the task specifications: x_d(t)
Configure the adaptive controller and the model learning procedure
for n = 1 → N_episodes do                                   ▷ For each episode
    for t = 0 → T do                                        ▷ For each time-step
        Get ẍ_r from Eq. 4.8
        Compute the cost function for the QP given the ẍ_r
        Get τ from the QP with the updated cost function and the learned model h*(q, q̇, q̈)
        Apply τ to the robot and collect data
        Update the adaptive controller with Eq. 4.10
    end for
    ID model learning (Section 4.3.2)
end for
```

Practical Considerations

Gaussian Process Regression has a training time complexity of $O(n^3)$ and is thus impractical when having to deal with many samples. Since our controllers operate at high frequency (around 200 Hz in our experiments), we can easily gather big datasets. There are many approaches to approximately learning GPs that reduce the time complexity (Quinonero-Candela and Rasmussen (2005)), but we chose to just subsample the data in order to reduce the number of points. We performed initial experiments in the simulated environments, and we did not observe any significant deterioration of the performance of the SCQP algorithm compared to when using all the points. On the other hand, when using all the points we could not manage to achieve real-time querying of the GPs for realistic QP controllers.

Another important point that we considered is the careful mixing of the prior model and the error model learned by the GPs. The QP controllers are model sensitive and can easily fail if there are inconsistencies. For this reason, we did not optimize the hyper-parameters of the GPs so we could consistently fall back to the prior model away from data points. Initial experiments with hyper-parameter optimization frequently led to QP optimization failures.

Conversely, the GPs without hyper-parameter optimization rarely triggered QP failures.

We use the limbo C++11 library for GP regression and the GP derivatives (Cully et al. (2018)).

4.4 Simulated Experiments

We aim to answer the following questions with our simulations:

(i) Can the SCQP method cope with big model mismatches?

(ii) Can the SCQP method generalize to several different scenarios and robot setups?

(iii) Can the SCQP method generalize to high-dimensional robots (e.g., humanoids) and contact-rich tasks?

(iv) Is learning the ID alone, or using taskspace adaptive control alone enough?

To answer these questions, we devise the following scenarios:

(i) A 7-DoFs KUKA LBR iiwa manipulator (14kg version) that tracks end-effector trajectories with an unknown mass attached to the end-effector.

(ii) A 32-DoFs PAL Robotics Talos humanoid robot[4] that performs a waving task while having unknown masses attached to both of the hands and/or unknown friction coefficients.

(iii) Two 7-DoFs KUKA LBR iiwa manipulators (14kg versions) that coordinate in order to manipulate a box with unknown mass; we provide preliminary results for this scenario.

For the above scenarios (except the experimental bimanual task), we experiment with the following approaches:

(i) Our SCQP approach (see Section 4.3.2)

(ii) Learning the ID model (as in Section 4.3.2) combined with a PID end-effector controller (the case where the adaptive controller is absent)

(iii) Using just the adaptive controller without any model learning

If not stated otherwise, we control the robot(s) at 200 Hz and each episode has a length of 10 s. We utilize the DART simulator (Lee et al. (2018)) with the robot_dart wrapper[5]. Note that all robots used in the thesis can be controlled directly in torque mode, i.e., we can send the computed joint torques as the direct command for the robot actuators.

[4] We disable the hands and use the 30-DoFs.
[5] https://github.com/resibots/robot_dart/

4.4.1 KUKA LBR iiwa Trajectory Tracking

We begin the evaluation of our methods using a 7-DoFs KUKA iiwa manipulator that needs to track specified end-effector trajectories. We will perform two types of experiments: (a) a task that involves moving the end-effector only along one axis (z-axis/gravity direction) and (b) a task that involves moving the end-effector along two axes (the yz-plane). We perform these two tasks to extensively evaluate our method against baselines. In both scenarios, to emulate real-world model uncertainties, we consider the following model mismatches:

(i) the QP controller assumes perfect actuators (no Coulomb friction or damping), whereas in the real world the actuators have both Coulomb friction and damping; and

(ii) a 1 kg mass attached to the end-effector of the actual KUKA that is not presented to the QP model.

The first mismatch represents the typical differences between the *ideal* and the *actual* model of the actuators. The second mismatch has a significant effect on the dynamics of the robots and simulates an exaggerated case of an unexpected situation. Imagine the case, for instance, where the robot is lifting an object, and abruptly, the force-torque sensor fails and outputs zero wrenches at the end-effector. Lastly, we add noises in the joint position and velocity measurements, emulating noisy real-sensor feedback.

QP Configuration

Here, we configure the QP with the desired Cartesian acceleration of the end-effector given by:

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}. \tag{4.18}$$

Thus, the QP formulation can be written as follows:

$$\begin{aligned} \boldsymbol{H}_{\text{acc}} &= [\boldsymbol{J}(\boldsymbol{q}) \;\; \boldsymbol{0} \;\; \boldsymbol{0}] \\ \boldsymbol{b}_{\text{acc}} &= \ddot{\boldsymbol{x}}_r - \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}. \end{aligned} \tag{4.19}$$

Finally, we add regularization constraints such as joint position and velocity limits, preferred null joint configurations, actuator torque limits, etc. Hence the optimization variables do not violate the functional limits and the robot remains stable.

Z-axis Tracking

The objective here is that the robot's end-effector has to follow a sinusoidal trajectory on the z-axis. The results showcase that when using the SCQP, the learning converges faster (i.e., in fewer trials/episodes) and to a more accurate model than when using the baselines; see Figure 4.2. The PID controller, despite being tuned with high gains, yields to poor tracking with a high

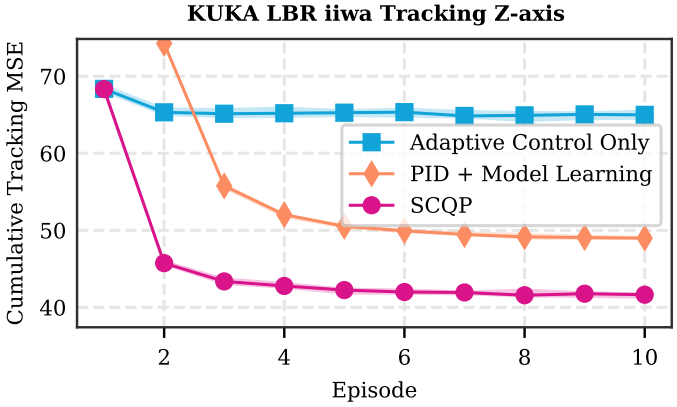


Figure 4.2: Results of the simulated experiment with KUKA iiwa, z-axis tracking. The experiment is repeated 20 times, each consisting of 10 episodes in succession. The tracking error for an episode is the cumulative mean square tracking error over time-steps. Solid lines are the median over 20 replicates and the shaded regions are the regions between the 5th and 95th percentiles.

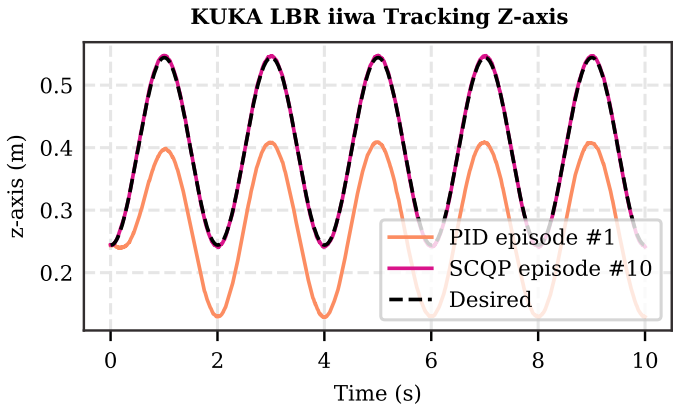


Figure 4.3: KUKA iiwa z-axis tracking trajectories: the first episode with PID and wrong QP model where the model mismatch affects the tracking, and the last episode with the SCQP approach which is able to track the desired trajectory.

cost. This is because the model mismatch is, indeed, significant, and the PID controller cannot compensate for it without the aid of the learned ID model. The adaptive controller achieves a relatively better cost even in the initial trials, and the tracking performance, without the learned model, improves over the remaining trials. However, it is inadequate to compensate for all the unmodeled dynamics, for which it needs the learned ID model. In other words, we observe that the model flexibility issue in fast online learning with adaptive controller is tackled by learning the residual dynamics in SCQP.

Qualitatively, the SCQP approach can quickly compensate for model mismatches and requires less than 5-6 episodes to achieve a desirable trajectory tracking. Figure 4.3 showcases typical z-axis trajectories before and after learning.

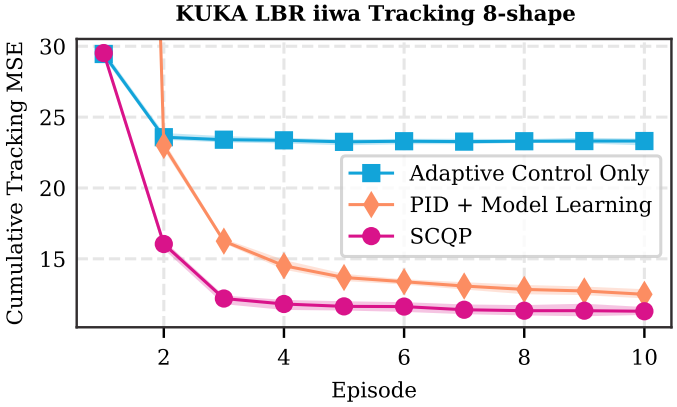


Figure 4.4: Results of the 8-shape trajectory tracking experiment on KUKA iiwa in a simulated environment. The experiment is repeated 20 times, each consisting of 10 episodes in succession. The tracking error for an episode is the cumulative mean square tracking error over time steps. Solid lines are the median over 20 replicates and the shaded regions are the regions between the 5th and 95th percentiles.

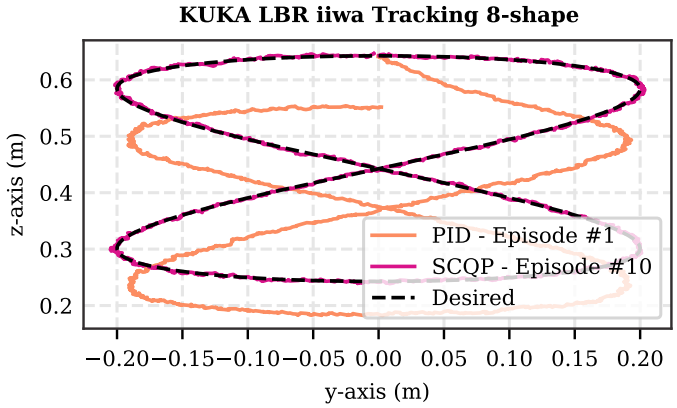


Figure 4.5: KUKA iiwa yz-axis tracking: typical trajectories across episodes with PID and the SCQP approach. SCQP enables accurate tracking of the desired trajectory.

YZ-plane Tracking

In this task, the robot's end-effector has to follow an 8-shaped trajectory in yz-plane. The results closely follow the outcomes of the previous scenario (Figure 4.4). The SCQP approach converges in fewer episodes than both baselines. Model learning with a PID end-effector controller can achieve good results but requires more episodes to converge. The adaptive controller alone cannot sufficiently compensate for all the model mismatches.

Qualitatively, the SCQP approach can quickly compensate for model mismatches and requires less than 5-6 episodes to achieve desirable trajectory tracking. Figure 4.5 illustrates typical yz-plane trajectories of the SCQP and the PID+model learning approaches.

4.4.2 Talos Humanoid Task

Here, we control a 32-DoFs Talos humanoid robot. While maintaining balance, the robot has to follow a sinusoidal trajectory with the right arm and keep the left arm in place; see Figure 4.6, left. We devised this scenario to evaluate our approach in high-dimensional state/action spaces with a more complex QP task. In addition to considering, for each arm, the same mismatch models as those in Section 4.4.1, we assume that the friction coefficients of the contact surfaces are not known. The real world has a coefficient of friction set to 0.7 whereas the QP model takes a coefficient of 1, i.e., contact surfaces are more slippery than what the QP expects.

This assumption attempts to create a model mismatch in a crucial part of the environment, strongly affecting the stabilization of the humanoid robot. Lastly, we add noise to the joint position and velocity measurements, emulating real sensory feedback.

QP Configuration

We define six Cartesian acceleration tasks following Eq. 4.19: (i) one tracking task per arm end-effector, (ii) two tracking tasks for the torso (COM position and upright preference), and (iii) one zero acceleration task (without feedback) for each foot. We also define one 6D contact constraint per foot to handle the balance of the humanoid (we assume the contact points are in the middle of the feet). More precisely, for each foot, we define a contact as inequality constraints of the following form:

$$\begin{aligned} \boldsymbol{H}_{I\text{contact}} &= [\mathbf{0}\ \ \mathbf{0}\ \ \boldsymbol{C}] \\ \boldsymbol{b}_{I\text{contact}} &= \begin{bmatrix} -\infty & -\infty & 0 & 0 & F_{min} \\ 0 & 0 & \infty & \infty & F_{max} \end{bmatrix} \end{aligned} \tag{4.20}$$

where $\boldsymbol{C}$ is defined as:

$$\boldsymbol{C} = \begin{bmatrix} (-\mu\boldsymbol{n} + \boldsymbol{t}_1)^T \\ (-\mu\boldsymbol{n} + \boldsymbol{t}_2)^T \\ (\mu\boldsymbol{n} + \boldsymbol{t}_1)^T \\ (\mu\boldsymbol{n} + \boldsymbol{t}_2)^T \\ \boldsymbol{n}^T \end{bmatrix} \tag{4.21}$$

and where $\mu \in \mathbb{R}^+$ is the coefficient of friction, $\boldsymbol{n} \in \mathbb{R}^3$ is the contact normal, and $\boldsymbol{t}_i \in \mathbb{R}^3$ are the tangential directions. We also add a few rows for constraining the center of pressure similar to (Feng et al. (2014)). Instead of using a center of pressure constraint, one can use four contact points in the corners of the foot.

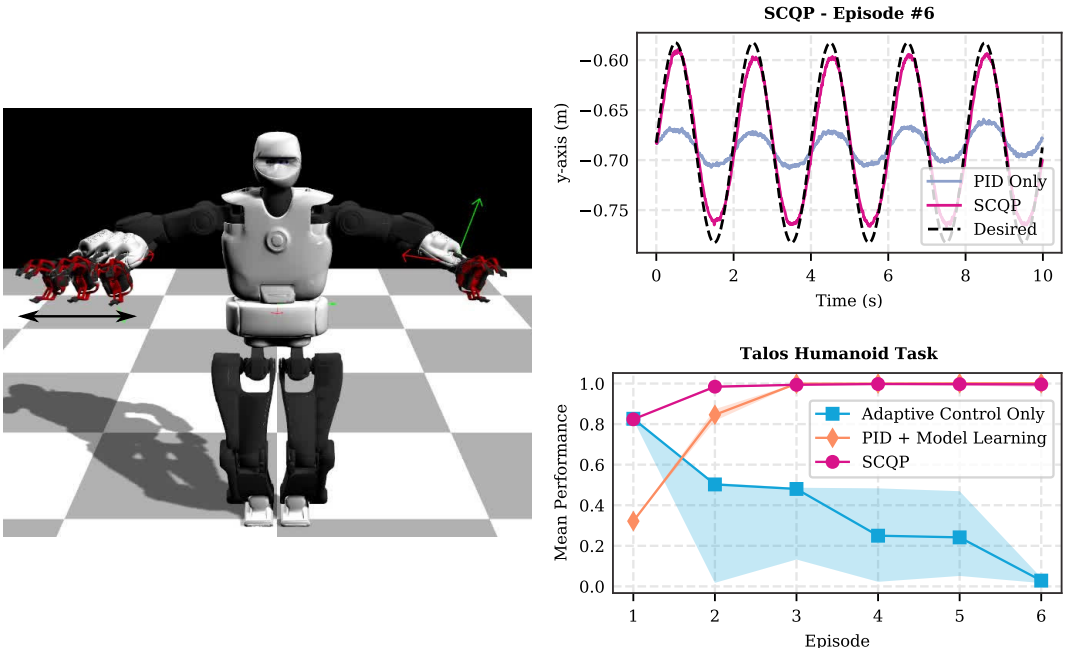


Figure 4.6: Successful trial of the Talos task. Through ID learning and online adaptation, SCQP learns to perform the waving task (left) without falling despite relatively big mismatches in the mass of the arms and the friction coefficient. The phase tracking accuracy during the waving task (top right) indicates the effectiveness of model learning in SCQP after only a few trials compared to the other approaches (bottom right). Solid lines are the median over 10 replicates and the shaded regions are the areas between the 25th and 75th percentiles.

Results

The results show that our SCQP method can scale to high-dimensional systems and handle passive contacts; see Figure 4.6. The SCQP converges faster (i.e., in fewer trials/episodes) to high performance controllers than the baselines. In this scenario, the adaptive controller alone might diverge if not properly tuned (due to the smaller value of the coefficient of friction). Thus, to showcase this drift issue and also the importance of the learned ID model, we chose a parameter configuration in which the adaptive controller alone diverges. Then, we used the same parameters for SCQP where model learning is present. The gains of the PID controller, however, are tuned to get its best performance for maintaining the robot's stability. In Figure 4.6 (right), we observe that the learned model "stabilizes" the adaptive controller that would diverge if left on its own.

Qualitatively, the SCQP approach is able to quickly compensate for model mismatches and requires less than 3-4 episodes to achieve desirable trajectory tracking. Figure 4.6 (middle) showcases a typical y-axis trajectory. The supplementary video shows an example of the learning procedure[6].

[6]The video is also available at https://youtu.be/cA-_SKoO_9c.

4.4.3 Preliminary Experiments on Bimanual Manipulation

In this scenario, we perform a bimanual manipulation task in which two KUKA LBR iiwa arms have to lift a box of unknown mass; see Figure 4.7. This experiment is used to simulate contact-rich tasks, and tasks that objects need to be manipulated. We consider the following model mismatches that act in combination (similar to the previous):

(i) the QP controller assumes perfect actuators (that is, no Coulomb friction or damping), whereas in the real world the actuators have both Coulomb friction and damping; and

(ii) the mass of the box is not well-calibrated and there is a mismatch of 0.5 kg (the real box has a mass of 1.5 kg whereas the QP model assumes 1 kg).

QP Configuration

To control the robots while performing the bimanual task, we extend the decision variables to contain the acceleration and torques of all entities, i.e., the two robots and the box. We add one constraint for the dynamics of each entity; see Eq. 4.2. To "connect" the two arms with the box, we have two sets of contact forces for the contact between each arm and the box. We then couple the dynamics of the individual entities by inserting the forces at the correct places, i.e., $\boldsymbol{J}_{arm}^{T}\boldsymbol{W}$ for the arms and $-\boldsymbol{J}_{box}^{T}\boldsymbol{W}$ for the box, since the forces acting on the box are identical in magnitude and in the opposite direction to the actions of the ones on the arms.

Results

We provide preliminary results of this contact-rich task that involves active contacts for manipulation. The results illustrate that our SCQP approach can be used to learn this type of task. The most challenging part is learning the ID model. We were able to consistently learn a good model within three episodes, but after the 3rd episode our model learning pipeline did not work consistently: we obtained big mean square errors in the training set meaning that something did not go well with the model learning. Nevertheless, three episodes were enough for the SCQP approach to improve the performance and produce trajectories that achieve the desired box movement. An example trajectory can be seen in Figure 4.7 as well as in the supplementary video.

4.5 Physical Robot Experiments

In order to validate our SCQP approach in the physical world, we devise two setups. The first is similar to Section 4.4.1, where a KUKA manipulator needs to track a desired trajectory. The second setup demonstrates the SCQP application in a pick-and-place task using a robotic hand while both the hand's and objects' dynamics are unknown. In both setups, apart from the imposed mass mismatch there is also the "reality-gap" in this scenario since many small

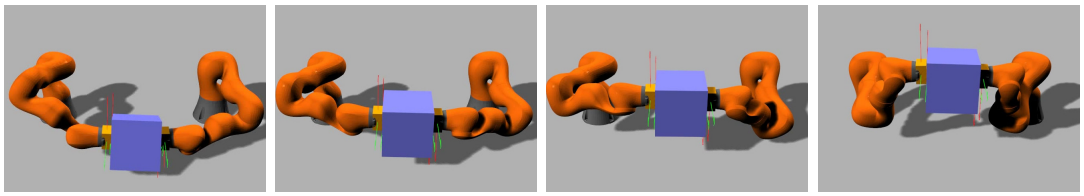


Figure 4.7: From left to right, screenshots of a successful trial of a bimanual manipulation task: grasping and manipulating a box with unknown mass. Through ID learning and online adaptation, SCQP learns to insert higher contact force to avoid slippage. Also, thanks to the torque control scheme, the robot configuration varies during task execution to maintain compliant joint behavior.

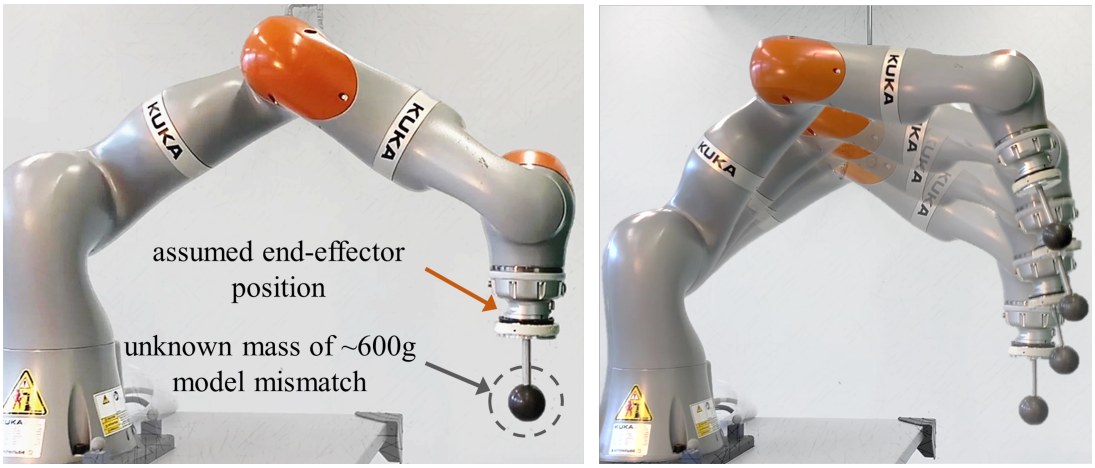


Figure 4.8: Physical robot setup with a mass mismatch at the end-effector. The KUKA manipulator needs to track a desired trajectory. This task is similar to Section 4.4.1 but is here applied to a real robot where real-time control as well as overcoming joints' friction and nonlinearities add to the control problem.

yet important details are not modeled in our original QP model (e.g., idealistic actuators). The robot is controlled at 200 Hz and each episode lasts 10 s.

4.5.1 Tracking Periodic Trajectory on Z-axis

As in the simulated experiment in Section 4.4.1, the robot has to follow a periodic trajectory on the z-axis with a mass mismatch at the end-effector (around 0.6 kg mismatch, see Figure 4.8). The results show that SCQP works in a physical system and provides similar performance to the simulated variant; see Figure 4.9. In particular, the SCQP converges to low-error trajectory tracking in less than $30-40$ s of interaction time (3-4 episodes); see Figures 4.9 and 4.10. Additionally, it performs better than the PID+model learning baseline, whereas it performs comparably to the adaptive control alone. Given the lower variance in tracking error (over 10 replicates), the SCQP shows higher consistency and robustness in tracking compared to the

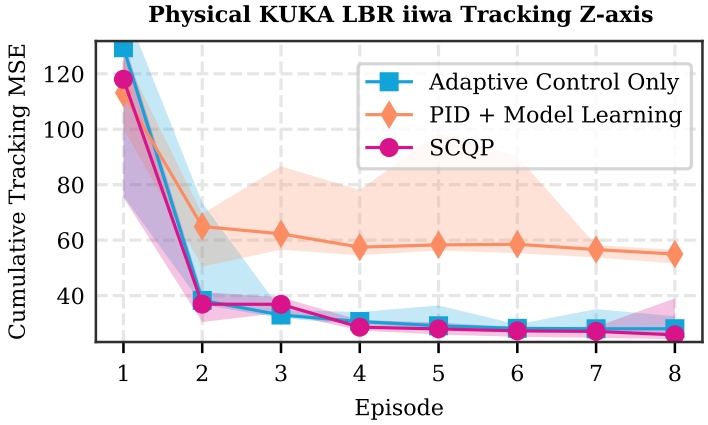


Figure 4.9: Results of real KUKA iiwa experiments with end-effector mass mismatch in the z-axis tracking. The experiment is repeated 10 times, each consisting of 10 episodes in succession. The tracking error for an episode is the cumulative mean square tracking error over time steps. Solid lines are the median over 10 replicates and the shaded regions depict the region of the 5th to 95th percentiles.

baselines. Qualitatively, the SCQP approach can quickly compensate for model mismatches and requires less than 5-6 episodes to achieve desirable trajectory tracking robustly. The supplementary video shows an example of the learning and control procedure.

Qualitatively the SCQP approach is able to quickly compensate for model mismatches and requires less than 5-6 episodes to achieve desirable trajectory tracking robustly. The supplementary video shows an example of the learning and control procedure.

4.5.2 Pick-and-Place with a Robotic Hand

In the experiment, we perform a pick-and-place scenario with a robotic hand. The task is to grasp different objects from a certain place and drop them into a bucket fixed in a different position. The robot has to follow a specific trajectory while holding the object vertical to the xy-plane; see Figure 4.11. In this experiment, the dynamics and physical properties (e.g., mass and inertia) of the robotic hand and the objects are unknown (the robotic hand is around 1.5kg and the objects are 250 ± 40g). We use objects of different weights for the task to showcase that SCQP can account for online uncertainties in addition to residual model learning. The experiment is an extreme use case of SCQP where it knows nothing about the hand and the objects to be grasped (a rather unrealistic assumption since we usually have some idea about the properties of the hand and the objects). Figure 4.11 shows that SCQP is capable of executing the task successfully despite significant unmodeled dynamics at the end-effector and model changes from one object to the other. Initially, the robot drastically deviates from the desired trajectory; however, after residual model learning, the tracking error reduces progressively. The supplementary video shows successful and failed examples of task execution for this experiment.

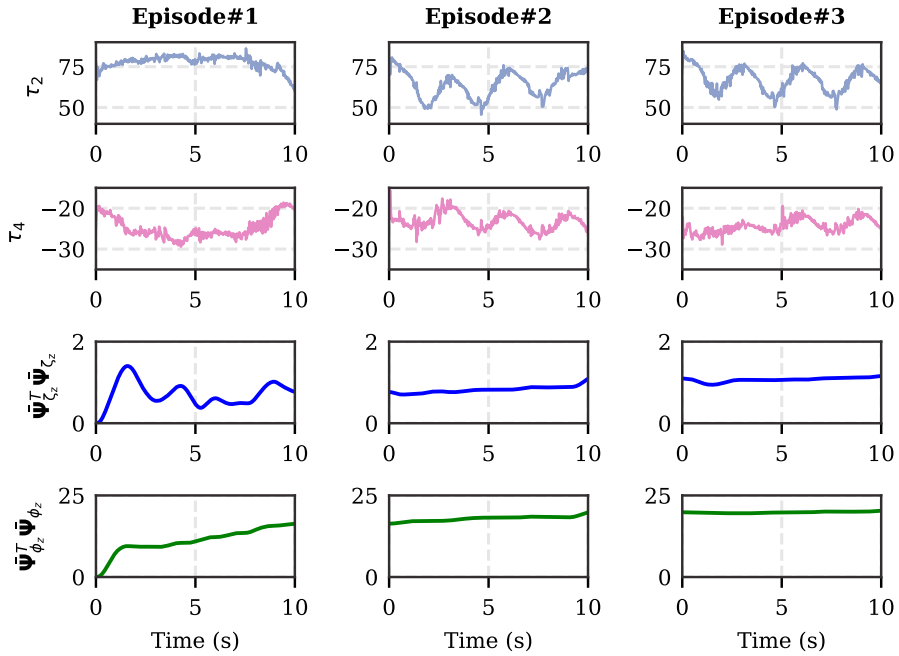


Figure 4.10: Examples of control responses and adaptive gains during the first three episodes (see Figure 4.9) of the periodic trajectory tracking task. The first two rows are the second, τ_2, and the fourth, τ_4, joint torques in $N.m.s^{-1}$, respectively. These two joints are the most load-bearing in task execution. The third and the fourth rows are the norm of linear, $\bar{\Psi}_{\zeta_z}^T \bar{\Psi}_{\zeta_z}$, and nonlinear, $\bar{\Psi}_{\phi_z}^T \bar{\Psi}_{\phi_z}$, adaptation gains active on the z-axis, respectively. After the first episode, the learned residual dynamic model results in modifying the torque commands to achieve a higher tracking accuracy in a consistent fashion; see Figure 4.9. Also, adaptive gains converge to constant values, and since big model mismatches are handled by the residual model learning, the online adaptation remains reactive to uncertainties faced online.

4.6 Discussion and Summary

In this work, we proposed a novel combination of an MRAC scheme with an ID model augmented QP-controller. Our main intuition was to merge a fast online adaptive control law in taskspace to regulate the cost function of the QP with a slower ID model learning procedure inserted inside the QP model. This pipeline, called SCQP, was effective, and we could successfully apply it to many different scenarios and robots.

In particular, the SCQP was able to compensate, in a handful of trials, for large model inaccuracies in tasks ranging from simple end-effector tracking to humanoid balancing and contact-rich tasks. Using SCQP, one can avoid the tedious tuning of PID controllers while also capturing significant unmodeled dynamics with the learned ID model.

Despite the successful application of SCQP, there remain several limitations that we would like to address in future: (a) SCQP requires either a different model for each contact configuration of the system or a learning model that can generalize to different contact configurations, and

(b) learning the ID model is itself a difficult task (as we observed in Section 4.4.3). Although there exist methods for learning ID models with contacts (Calandra et al. (2015)), learning effectively ID models from unstructured data[7] remains an open problem that deserves further investigation.

Finally, in this work, we assumed that the high-level tasks remained fixed throughout the process. It is straightforward to combine SCQP with methods that update online the high-level planning part. Task modulating can be crucial if the original specifications are not achievable; for example, the left arm is damaged and the task has to be performed with the right arm. Moreover, our approach can be combined with reinforcement learning algorithms. The idea here is to enable exploration around the commands the QP outputs, while ensuring safety and not allowing large deviations that could potentially harm the robot.

[7] We do not have access to a full static dataset, but rather collect data as we apply the controller on the system.

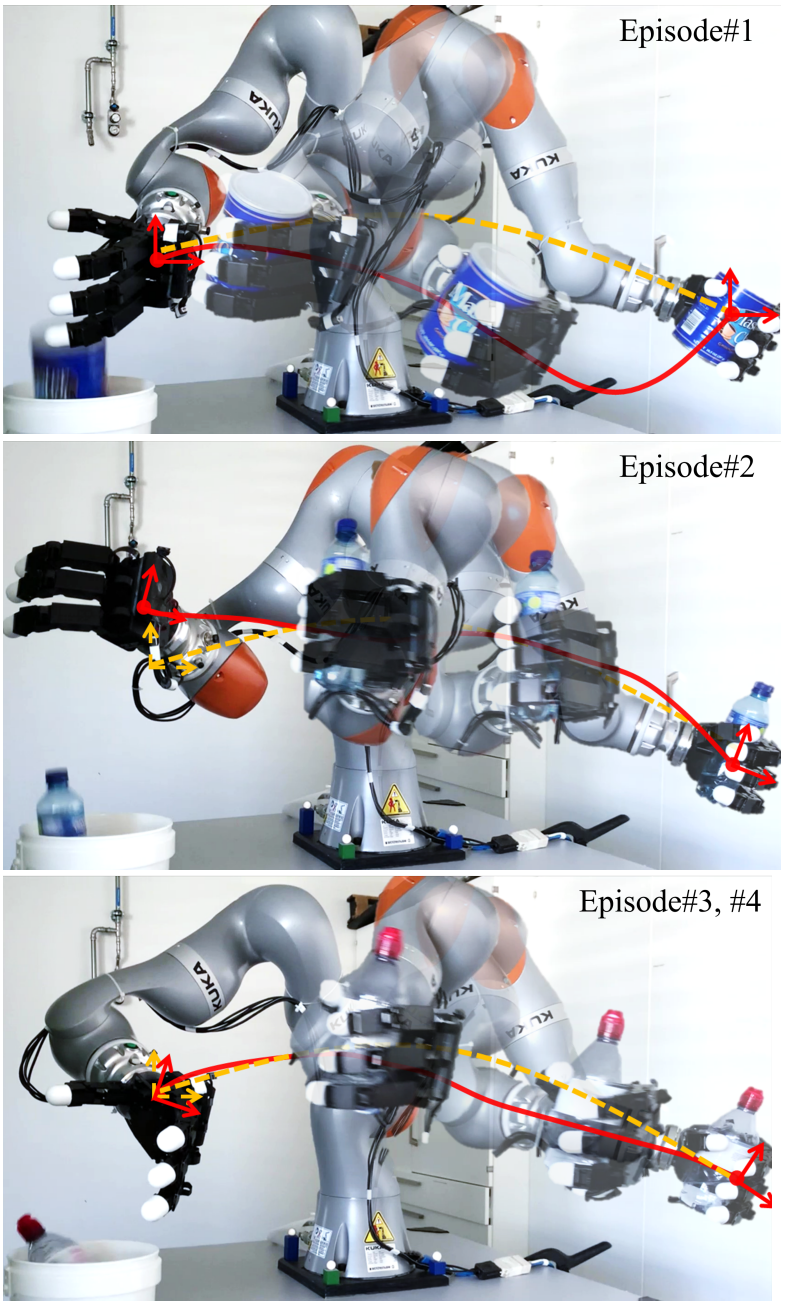


Figure 4.11: Examples of pick-and-place task with a robotic hand, Section 4.5.2. TThe robot first grasps an object, follows a specific trajectory (dashed yellow line) while keeping the object vertical, and places the object into a fixed bucket. The dynamics and physical properties of the robotic hand and the objects are unknown (the robotic hand is around 1.5kg, and the objects are 250 ± 40g), and different objects are used for each episode. In the first episode, the robot has a significant deviation (solid red line is the taken trajectory) from the desired trajectory; however, after residual model learning, the tracking error reduces progressively in the other episodes. In the third episode, the robot fails to drop the object in the target place; nevertheless, the task is successfully completed in the fourth episode with the same object.

5 Adaptive Fingers Coordination for Robust Grasp and In-Hand Manipulation

We present a control framework for achieving a robust object grasp and manipulation in hand. In-hand manipulation remains a demanding task as the object is never stable and task success relies on carefully synchronizing the fingers' dynamics. Indeed, fingers must simultaneously generate motion while maintaining contact with the object and, by staying within the hand's frame, ensuring that the object remains manipulable. These challenges are exacerbated once the hand gets disturbed or when the internal dynamics of the manipulated object are unknown, such as when it is filled with liquid moving during manipulation.

In this part of the thesis, we address the problem of ensuring robust in-hand manipulation when faced with a poor model of the object's dynamics, model imperfections, and external disturbances. We hypothesize that the key to robust manipulation is to carefully synchronizing the fingers' dynamics and offer a novel control strategy based on coupled dynamical systems (DSs), combined with an adaptive torque-controller. This controller provides live adaptation of the position and force generated by the fingers to stabilize the object. Additionally, we propose a joint-level adaptive torque controller to track the fingers' desired trajectory generated by the DS and to regulate joint-impedance gains.

In summary, we contribute to the problem of robustness in grasp and in-hand manipulation by:

(i) Achieving a robust coordinated multi-finger system for grasp and manipulation. We propose a novel coupled dynamical system that retains the coupling of the fingers in the face of perturbations.

(ii) Combining MRAC with joint impedance regulation to adaptively control joint torques in real-world grasp and manipulation tasks.

In addition to these two contributions, we make one minor one: We learn task segments of in-hand manipulation from human demonstration that, to the best of our knowledge, is conducted for the first time in this study and will be discussed later in *Section 5.6.4*.

At the time of writing, the work presented in this chapter has been accepted and is under the publication process as *Khadivar, F., and Billard, A. "Adaptive Fingers Coordination for Robust Grasp and In-Hand Manipulation under Disturbances and Unknown Dynamics." IEEE Transactions on Robotics (T-RO), 2022.*

5.1 Introduction

In-hand manipulation is the ability to re-position an object within a hand. Grasping and manipulating objects of more variety and complexity is best explored when using a robotic hand with multiple degrees of freedom (Dafle et al. (2014)).

Our solution to coordinate fingers is based on dynamical systems. The coupled DSs we reportedly used to synchronize multiple robots' dynamics Chung and Slotine (2009); Caccavale and Uchiyama (2016); Mirrazavi Salehian et al. (2018); Kastritsi et al. (2018)). We extend this concept to in-hand manipulation. When using a coupled DS, the challenge is to enable the hand to produce the desired motion on the object. We achieve such planning for fingers' synchronization by introducing a novel approach that controls the coupling of DSs of multi-limbs. More precisely, we design an intermediate dynamics that regulates the relative speed of the fingers to accelerate or decelerate their motion. Through this intermediate DS, we ensure that fingers apply, in coordination, the desired motion and force on the object. We obtain a closed-form expression that enables automatic and synchronous re-planning and repositioning of the fingers in response to changes in the object's pose.

Compared to coupling DS with a virtual reference trajectory (Mirrazavi Salehian et al. (2018)) or using temporal scaling dynamic (Kastritsi et al. (2018)), our intermediate DS coupling method ensures, in addition to being robust to disturbances to each finger, that all the fingers follow a specific desired trajectory, leading the object to the desired pose. For instance, though all fingers are synchronized to move together, if one finger is disturbed, the others wait for the perturbed finger to re-synchronize, rather than retracting or deviating from the intended trajectory; see Figure 5.1 for an illustration.

The difficulties with real-world execution, task planning complexity, and feedback control law inefficiency are primarily due to imprecise knowledge of object properties, contact mechanics, and the robot's dynamics (Bicchi (2000)). In this respect, considerable efforts have been put into designing control systems based on either hybrid force/position schemes or varying-impedance modulation; see a thorough review by Ozawa and Tahara (2017). The complexity in modeling hand-object dynamics has encouraged some researchers to investigate data-driven algorithms that, as an alternative, use approximate models, such as model-based (Kumar et al. (2016)) or model-free (Rajeswaran et al. (2017); Andrychowicz et al. (2020); Zhu et al. (2019)) reinforcement learning. For such learning algorithms, where the need for a perfect simulated environment is substantial, extracting appropriate features of physical interaction is a significant burden that leads to all the issues related to sim-to-reality and vice versa. Furthermore, data-driven approaches cannot cope easily with perturbations unless they have

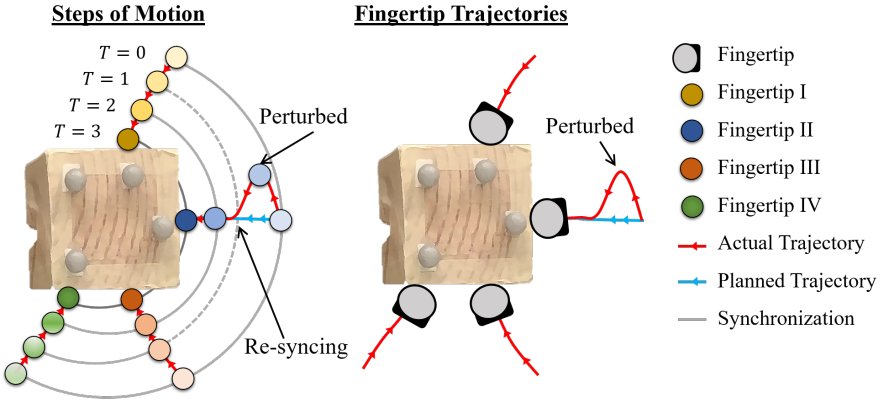


Figure 5.1: A synchronized grasp example. All fingers are coordinated to move together. If one finger (here fingertip II) is perturbed anytime during motion, the others wait for it to re-synchronize ($T = 0$ to $T = 2$), rather than retracting or deviating from the intended trajectory. The perturbed finger, also, accelerates to synchronize while recovering the planned trajectory and approaching the object with correct orientation.

seen these as examples during training.

The adaptation of the torques online is key to mitigating uncertainties arising from poorly modeled robot dynamics, object-finger interactions, and the object's physical properties (change in mass distribution). The concept of adaptive controllers (Goodwin and Sin (2014)) per se is well known in robotics. From the rich pool of adaptive controllers, we borrow a model reference adaptive control (MRAC) (Ortega and Spong (1989)) and tailor it for our coordinated task-planning method. To the best of our knowledge, our work is the first to utilize MRAC as low-level torque control for grasp and manipulation, in a real robotic setup. Furthermore, to better suit the controller for the task at hand, we embed an impedance regulation to the adaptive control to avoid the well-known issue of gains saturation.

Another common issue with in-hand manipulation is achieving the desired object pose within the reach of each finger and given the limited workspace of the hand. Throwing and re-grasping objects (Dafle et al. (2014); Chavan-Dafle and Rodriguez (2015, 2018)), utilizing palm and the extrinsic environment (Bai and Liu (2014)) are among the the proposed solutions. Dynamic forces on the object, if known precisely, can be utilized to manipulate the object as, for instance, in an impressive demo by Furukawa et al. (2006), in which the hand re-grasps a foam cylinder object by tossing it in the air and catching it. This type of manipulation largely relies on the dynamic properties and, in particular, on the object's moment of inertia. Re-grasping becomes nearly impossible, once the distribution of the object's mass is non-uniform or varying. The majority of the previous studies considered one-step in-hand manipulation (Shi et al. (2017); Sundaralingam and Hermans (2019); Pfanne et al. (2020)). Here, we learn manipulation sequence from human demonstrations and extract the coupling and the sequence across fingers motion. We show that embedding this into our controller offers immediate robustness to the above disturbances.

Rotating an object in hand is referred to as in-hand rolling that requires continuous finger gaiting (Han and Trinkle (1998)). For the most recent attempt at this task, researchers relied on learning the sequences of finger motion through reinforcement learning closing the loop in vision and haptic feedback (Xu et al. (2010); Rajeswaran et al. (2017); Andrychowicz et al. (2020)). In this work, we use the concept of learning human demonstrations to extract the coupling and the sequence across fingers motion. We show that embedding this into our controller offers immediate robustness to the above disturbances. Note that we perform various experiments, yet only the planning part of our framework differs among all the tests. Due to the adaptation properties, the low-level control is fixed across all experiments, reducing substantially the amount of engineering.

We analyze the closed-loop system and show that: (i) the coupled DS is asymptotically stable to the desired object's pose; (ii) convergence of the error of the adaptive torque-control; (iii) under no disturbances, the system provides torques to ensure force-closure for the grasp.

We evaluate the developed controller in four real-world robotic experiments: (a) comparing with control baseline and testing the hand controller in tracking tasks, (b) assessing the grasp adaptation with multiple objects of different dynamic properties, (c) for the same object, performing rotational and translational in-hand manipulation, and (d) rolling a cuboid in-hand by finger-gaiting. We perform the latter to demonstrate the strength of our approach in a challenging in-hand manipulation task.

5.2 Approach

We perform trajectory planning in taskspace and compute control commands in joint space. For planning, we devise our controller on the basis of dynamical systems as DS-based controllers are robust to perturbations (Khansari-Zadeh and Billard (2011)). More precisely, in our high-level planning, we express the task as a desired velocity $\dot{\boldsymbol{x}}_i^d \in \mathbb{R}^{d_x}$, with d_x being the dimension of position in taskspace, for each i-th finger following a DS of the form

$$\dot{\boldsymbol{x}}_i^d = f_i(\boldsymbol{x}_i) \tag{5.1}$$

$f_i(\cdot)$ is a continuous and asymptotically stable at an attractor $\boldsymbol{x}_i^* \in \mathbb{R}^{d_x}$ meaning that $\|\boldsymbol{x}_i - \boldsymbol{x}_i^*\| \rightarrow 0$ as $t \rightarrow \infty$. The function f can be linear, nonlinear, or cyclic, and it can hold and generate all possible trajectories to reach the target state (Khadivar et al. (2021b)).

In low-level control of a robot with n_j joints, we can set the joint positions, $\boldsymbol{q} \in \mathbb{R}^{n_j}$, or send joint torques, $\boldsymbol{\tau} \in \mathbb{R}^{n_j}$, as the control command. In practice, although position control is intuitive to implement, it relies on high gains that can, when contact arrives earlier than planned, damage both the robot's joints and the object. In grasp/manipulation applications, a precise knowledge of object shape and robot model (i.e., an accurate inverse dynamics models (Khadivar et al. (2021a)) is needed when using position control. Controlling joint torques is more suited to establishing compliant object-finger interactions, which leaves room

for adaptation to underlying imperfections. As a result, we follow a torque-based control approach for the low-level robot control:

$$\boldsymbol{\tau}_i = u_i(\dot{\boldsymbol{x}}_i^d, \boldsymbol{q}_i, \dot{\boldsymbol{q}}_i) \tag{5.2}$$

where $\boldsymbol{\tau}_i \in \mathbb{R}^{n_j}$ is the joint torques computed to realize the desired fingertip velocity $\dot{\boldsymbol{x}}_i^d$ from Eq. 5.1. Our objective in low-level control is to define the function $u_i(\cdot)$ for each i-th finger. With $u_i(\cdot)$, we adapt explicitly the joint compliance to allow handling uncertainties about the object's shape, mass distribution, and contact mechanics. In summary, we solve the grasp and manipulation problem in two steps:

(i) Coordinated planning in taskspace: We couple a group of DSs, $\{f_i(\cdot)\}_{i=1}^{n_f}$, to obtain the desired velocities, $\{\dot{\boldsymbol{x}}_i^d\}_{i=1}^{n_f}$, for all n_f fingertips. In *Section 5.3*, we synchronize fingers to achieve the coordinated motion required during grasp and in-hand manipulation.

(ii) Adaptive torque-control in joint space: We adapt joint torques and learn the mapping $u_i(\cdot)$ online, given the desired velocities $\dot{\boldsymbol{x}}_i^d$ from the previous step. Adaptation is based on MRAC and regulating joints' impedance, see *Section 5.4*.

5.3 Finger Synchronization Based on Dynamical Systems

We begin with a measure of relative distance variable and its intermediate DS. We derive the taskspace planner for each fingertip and describe how a single robot can be controlled with this intermediate DS. Next, we explain how multiple fingers are coupled through intermediate DSs, and how this results in coordination of fingers suitable for grasp and manipulation.

5.3.1 Intermediate Dynamic

Let's consider a single robot end-effector (fingertip in our case). The task is to control this end-effector to reach a *desired position* $\boldsymbol{x}^d \in \mathbb{R}^{d_x}$. An additional requirement is that $\boldsymbol{x}^d$ is needed to be between two specific points in the taskspace: $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)} \in \mathbb{R}^{d_x}$. These two points (see *Section 5.3.3*) are used for formulating the entire grasp and manipulation task. But, for now, let's assume that they are externally defined. Given this, the desired position of the robot, $\boldsymbol{x}^d$, can be expressed as a form of a relative distance measure:

$$\boldsymbol{x}^d = \boldsymbol{x}^{(2)} - z^d(\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}) \tag{5.3}$$

where $z^d \in \mathbb{R}_{[0,1]}$ is a variable representing the relative distance of the desired position, $\boldsymbol{x}^d$, from the second point $\boldsymbol{x}^{(2)}$. Instead of directly fixing z^d for the robot to track, we set an intermediate target, $\boldsymbol{x}^* \in \mathbb{R}^{d_x}$, defined by $z \in \mathbb{R}_{[0,1]}$:

$$\boldsymbol{x}^* = \boldsymbol{x}^{(2)} - z(\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}) \tag{5.4}$$

then regulating z to converge to z^d. This intermediate variable will later serve as the coupling variable for us. Determining and controlling z is the core of our coordinated planning problem. Regardless of the actual robot position, z is needed to evolve based on a dynamics that ensures $\|z^d - z\| \to 0$ as $t \to \infty$, hence we propose the following DS:

$$\dot{z} = g(z, z^d) = -\beta_1 \frac{1 - e^{-\beta_2(z - z^d)}}{1 + e^{-\beta_2(z - z^d)}} \tag{5.5}$$

both $\beta_1 \in \mathbb{R}^+$ and $\beta_2 \in \mathbb{R}^+$ are positive scalar values gearing the convergence behavior.

Theorem 1 *The DS given by (5.5) asymptotically converges to z^d i.e., $\lim_{t\to\infty} \|z^d - z\| = 0$.*

Proof *If we consider the Lyapunov function $V(z) = \frac{1}{2}(z - z^d)^2$, taking the time derivative results in $\dot{V} = (z - z^d)\dot{z}$. By replacing $\dot{z}$ with the proposed DS (5.5) we will have*

$$\dot{V} = -\beta_1 \frac{1 - e^{-\beta_2(z - z^d)}}{1 + e^{-\beta_2(z - z^d)}}(z - z^d)$$

for $z - z^d$ the derivative is $\dot{V}(z) = 0$, and for $z \neq z^d$, $\dot{V} = -\beta_1 \gamma(z)$ where $\gamma(z) = \frac{1 - e^{-\beta_2(z - z^d)}}{1 + e^{-\beta_2(z - z^d)}}(z - z^d)$. It is easy to check that for $z \neq z^d$, $\gamma(z) > 0$. If we take the derivative of γ:

$$\gamma' = \frac{1 - e^{-2\beta_2(z - z^d)} + 2\beta_2(z - z^d)e^{-\beta_2(z - z^d)}}{(1 + e^{-\beta_2(z - z^d)})^2}$$

then:

$$\begin{cases} \gamma' > 0 & z > z^d \\ \gamma' < 0 & z < z^d \end{cases}$$

hence for $z \neq z^d$, $\gamma(z) > 0$ and $\dot{V}(z) < 0$. As a result, the DS (5.5) is asymptotically stable and $\|z^d - z\| \to 0$ as $t \to \infty$.

Remark 1 *For any input value $z' \in \mathbb{R}_{[0,1]}$ to DS (5.5), we can find z by one step integration, i.e., $z = g(z', z^d)\delta + z'$, which sets the next intermediate target point, $\boldsymbol{x}^*$ (Eq. 5.4) for the robot to follow.*

5.3.2 taskspace Dynamical System

State-Dependent Target Point

Altering the variable z is equivalent to changing the target position, $\boldsymbol{x}^*$, for the robot. Accordingly, the attractor of the taskspace DS is state-dependent, which we need to consider in DS derivation for the robot end-effector. Given the robot end-effector position $\boldsymbol{x} \in \mathbb{R}^{d_x}$, variable

z, and the intermediate DS (5.5), we propose the following taskspace DS to control a robot end-effector:

$$\dot{\boldsymbol{x}}^d = f(\boldsymbol{x}, z) = -(z\mathbf{A}_x + \dot{z}\mathbf{I}_{d_x})(\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}) - \mathbf{A}_x(\boldsymbol{x} - \boldsymbol{x}^{(2)}) \tag{5.6}$$

$\mathbf{A}_x \in \mathbb{R}^{d_x \times d_x} \succ 0$ is the defined as the state matrix, and $\mathbf{I}_{d_x}$ is the identity matrix.

Theorem 2 *The taskspace DS given by (5.6) is asymptotically stable at the target point $\boldsymbol{x}^*$, i.e.,* $\lim_{t\to\infty} \|\boldsymbol{x}^* - \boldsymbol{x}\| = 0$.

Proof *We establish, based on the tracking error $\boldsymbol{x}_i - \boldsymbol{x}_i^*$, the Lyapunov function for the taskspace DS. Let's assume the Lyapunov function $V(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_i^*)^T\mathbf{P}(\boldsymbol{x}_i - \boldsymbol{x}_i^*)$ and take it's time derivative:*

$$\dot{V} = \frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_i^*)^T\mathbf{P}(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_i^*) + \frac{1}{2}(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_i^*)^T\mathbf{P}(\boldsymbol{x}_i - \boldsymbol{x}_i^*)^T.$$

By using DS (5.6) and the derivative of Eq. 5.4 into the time derivative of the Lyapunov function, we will have

$$\dot{V} = -\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_i^*)^T(\mathbf{A}_x^T\mathbf{P} + \mathbf{P}\mathbf{A}_x)(\boldsymbol{x}_i - \boldsymbol{x}_i^*)$$

which by satisfying similar conditions, such as Eq. 2.2, the stability of the DS (5.6) is guaranteed.

DS (5.5) is asymptotically stable at z^d meaning that $\lim_{t\to\infty}\|\boldsymbol{x}^d - \boldsymbol{x}^*\| = 0$, and DS (5.6) asymptotically converges to $\boldsymbol{x}^*$. Therefore, given Theorem 1 and Theorem 2, DS given by (5.6) is stable at the desired point $\boldsymbol{x}^d$, i.e., $\lim_{t\to\infty}\|\boldsymbol{x}^d - \boldsymbol{x}\| = 0$. Note that the stability of DS (5.6) at $\boldsymbol{x}^d$ is not necessarily asymptotic, see *Section 5.3.2*.

Feedback for Intermediate Variable

We obtain z by providing the DS (5.5) with the state feedback from the robot (see Remark 1). First, we compute the actual relative distance from the current position of the robot. To better distinguish the *actual relative distance* from the DS variable z, let's use α instead of z'. Let α be a measure of the current relative distance:

$$\alpha = \frac{(\boldsymbol{x}^{(2)} - \boldsymbol{x})^T(\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)})}{\|\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}\|^2} \tag{5.7}$$

then we can estimate z by one-step integration of the intermediate DS (5.5), $z = g(\alpha, z^d)\delta + \alpha$, for one time step δ. In this way, we use the intermediate DS in closed-loop control.

Overall DS for One Fingertip

The overall control loop of one fingertip finds the desired velocity, $\dot{\boldsymbol{x}}^d$, using the following DSs combined:

$$\begin{cases} \dot{z} = g(\alpha, z^d) \\ \dot{\boldsymbol{x}}^d = f(\boldsymbol{x}, z). \end{cases} \tag{5.8}$$

Different combination of these two DSs leads to different linear and nonlinear convergence behavior. Figure 5.2 illustrates $2D$ examples of the vector field for $\dot{\boldsymbol{x}}^d$ by using (5.8) where the intermediate dynamic, $g(\alpha, z^d)$, has a lower, an equal, and a higher convergence speed, compared to the robot taskspace dynamic, $f(\boldsymbol{x}, z)$. If a fingertip is perturbed while traveling from $\boldsymbol{x}^{(1)}$ to $\boldsymbol{x}^d$, from Figure 5.2, the slower intermediate DS will bring the finger back to the connecting trajectory of $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$. We take advantage of this property, both in grasp and manipulation planning: during grasping phase to ensure that the fingertips approach the object in the correct direction and, during manipulation, to reduce the deviation from the desired contact velocities.

5.3.3 DS Coupling and Coordination of Fingers

Coupling

To couple n_f fingertips, each controlled by DS (5.8), we define the intermediate DS for the i-th fingertip, $i = 1, \cdots, n_f$:

$$\begin{cases} \dot{z}_i = g(\alpha_i, z^c) \\ \dot{\boldsymbol{x}}_i = f(\boldsymbol{x}_i, z_i) \end{cases} \tag{5.9}$$

z^c acts as the *coupling variable*, and is:

$$z^c(\alpha_1, ..., \alpha_{n_f}, z^d) = \frac{1}{n_f + 1}(z^d + \sum_{i=1}^{n_f} \alpha_i) \tag{5.10}$$

z^c couples the entire system and captures the status of all fingers conveyed through the measurement of their actual relative distances $\{\alpha_i\}_{i=1}^{n_f}$.

Theorem 3 *The coupled system (5.9) for n_f fingers converges to a stable state at $\boldsymbol{x}_i^d$ for $i = 1, \cdots, n_f$, i.e., for each i-th fingertip as $t \to \infty$, $\|z^d - z_i\| \to 0$, and consequently, $\|\boldsymbol{x}_i^d - \boldsymbol{x}_i\| \to 0$.*

Proof *For each i-th finger, proving that $\lim_{t\to\infty} \|z^c - z_i\| = 0$ is similar to stability proof . Here, we want to show that if $\lim_{t\to\infty} \|z^c - z_i\| = 0$ then $\|z^d - z_i\| \to 0$, as well. From $z^c =$*

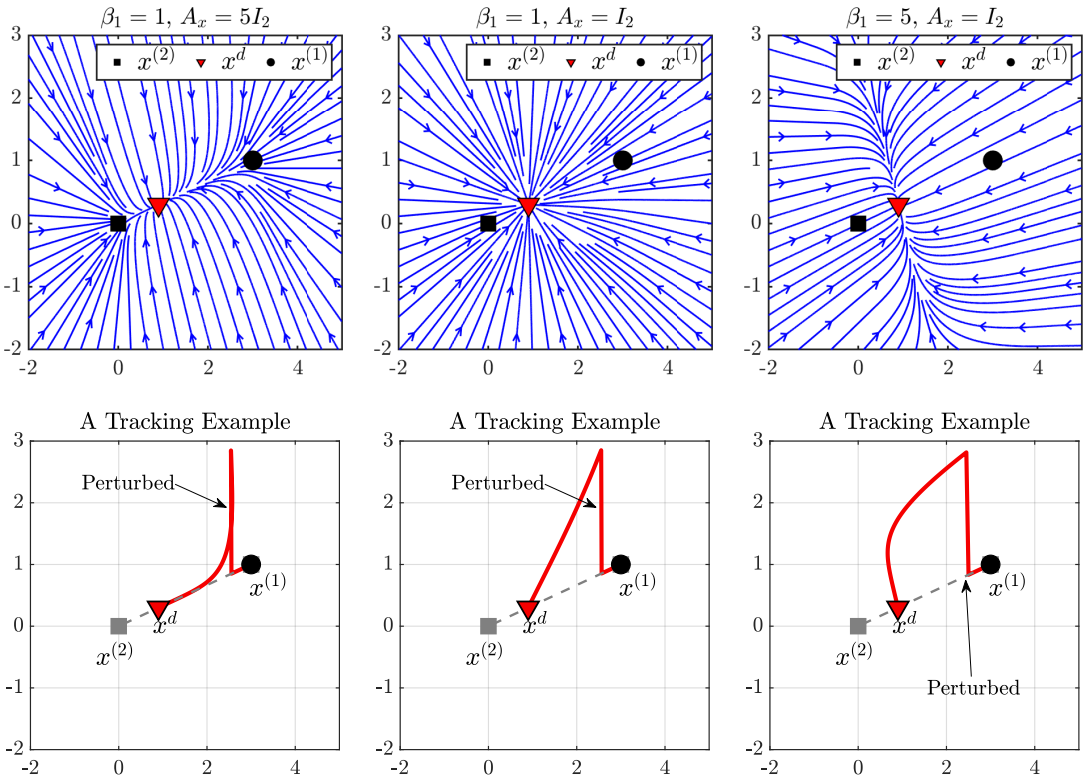


Figure 5.2: Examples of the vector field of the desired velocity, $\dot{\boldsymbol{x}}^d$,(right column) generated by DS (5.8), and examples of a perturbed tracking (left column). For each row of examples respectively, the cases illustrate the intermediate dynamic, $g(\alpha, z^d)$, that has a lower (top row), an equal (middle row), and a higher (bottom row) convergence speed, compared to the robot taskspace dynamic, $f(\boldsymbol{x}, z)$.

$\frac{1}{n_f+1}(z^d + \sum_{i=1}^{n_f} \alpha_i)$ *we can rewrite the error term* $z^c - z_i$ *in the following form:*

$$z^c - z_i = (z^d - z_i) - \frac{\sum_1^{n_f}(z^d - z_i)}{n_f + 1} \tag{5.11}$$

which means :

$$\sum_1^{n_f}(z^c - z_i) = \frac{1}{n_f+1}\sum_1^{n_f}(z^d - z_i) \tag{5.12}$$

by having the square of Eq. 5.11 and then the sum over the entire systems we will have:

$$\sum_1^{n_f}(z^c - z_i)^2 = \sum_1^{n_f}(z^d - z_i)^2 - \frac{n_f+2}{(n_f+1)^2}\Big(\sum_1^{n_f}(z^d - z_i)\Big)^2 \tag{5.13}$$

next we use the square Eq. 5.12 in the right hand side of Eq. 5.12 such that we acquire the

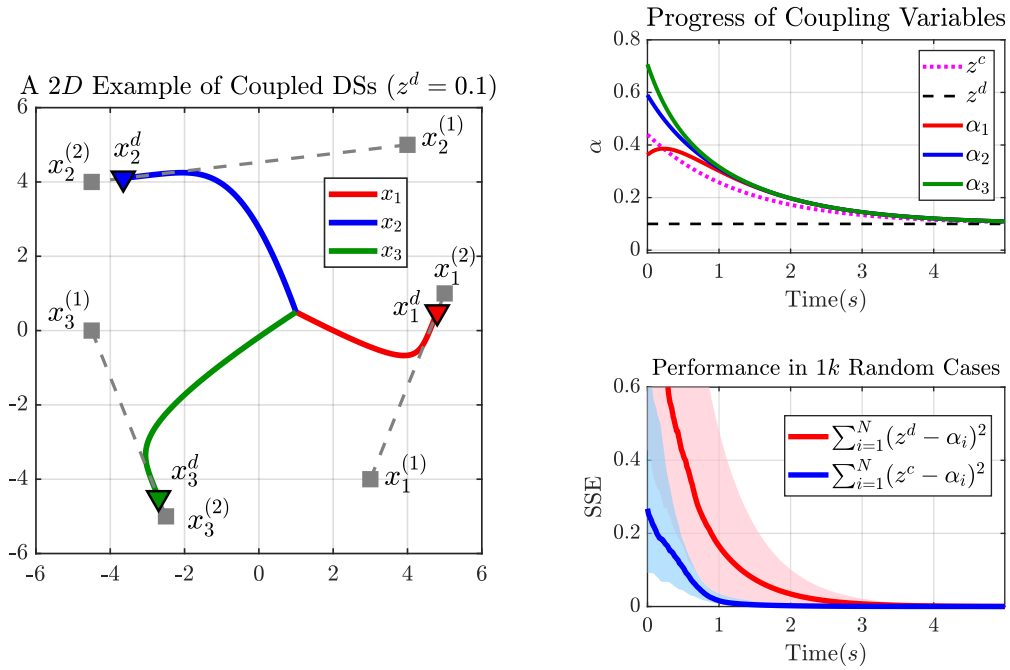


Figure 5.3: A coupled system of three robot end-effectors controlled by DS (5.9). The left figure demonstrates a tracking example, where all fingers start at the same initial position and are needed to reach their respective desired position $\boldsymbol{x}_i^d$, and where the top right figure is the progress of the corresponding coupling variables for this example. The bottom right figure shows the sum of square errors (SSE) in tracking the coupling relative variable z^c and the desired relative variable z^d in $1k$ cases of random initial positions and z^d. The solid lines are the median, and the shaded area displays 25-75 percentiles over $1k$ cases.

following equality:

$$\sum_{1}^{n_f}(z^d - z_i)^2 = \sum_{1}^{n_f}(z^c - z_i)^2 + (n_f + 2)\Big(\sum_{1}^{n_f}(z^c - z_i)\Big)^2 \tag{5.14}$$

which can also be observed in Figure 5.3. From Eq. 5.14, we conclude that when $\lim_{t\to\infty}\|z^c - z_i\| = 0$ *for all* $i = 1$ *to* n_f *then* $\lim_{t\to\infty}\sum_{1}^{n_f}(z^d - z_i)^2 = 0$, *and therefore,* $\|z^d - z_i\| \to 0$ *as* $t \to \infty$.

Figure 5.3 illustrates an example of three coupled DSs (5.9), where the robot end-effectors are initiated from the same position, $\boldsymbol{x}_1(t=0) = \boldsymbol{x}_2(t=0) = \boldsymbol{x}_3(t=0)$, and synchronously converge to their respective desired positions. From Figure 5.3 we also observe that in $1k$ random cases of initial positions and z^d, eventually $\|\alpha_i - \alpha_j\|_{i\neq j} \to 0$, $\|\alpha_i - z^c\| \to 0$, and $\|z^c - z^d\| \to 0$.

Defining $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$

The position of states $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ shapes the desired vector field and determines the behavior of the system in face of perturbation (see Figure 5.2). Defining where to place these two

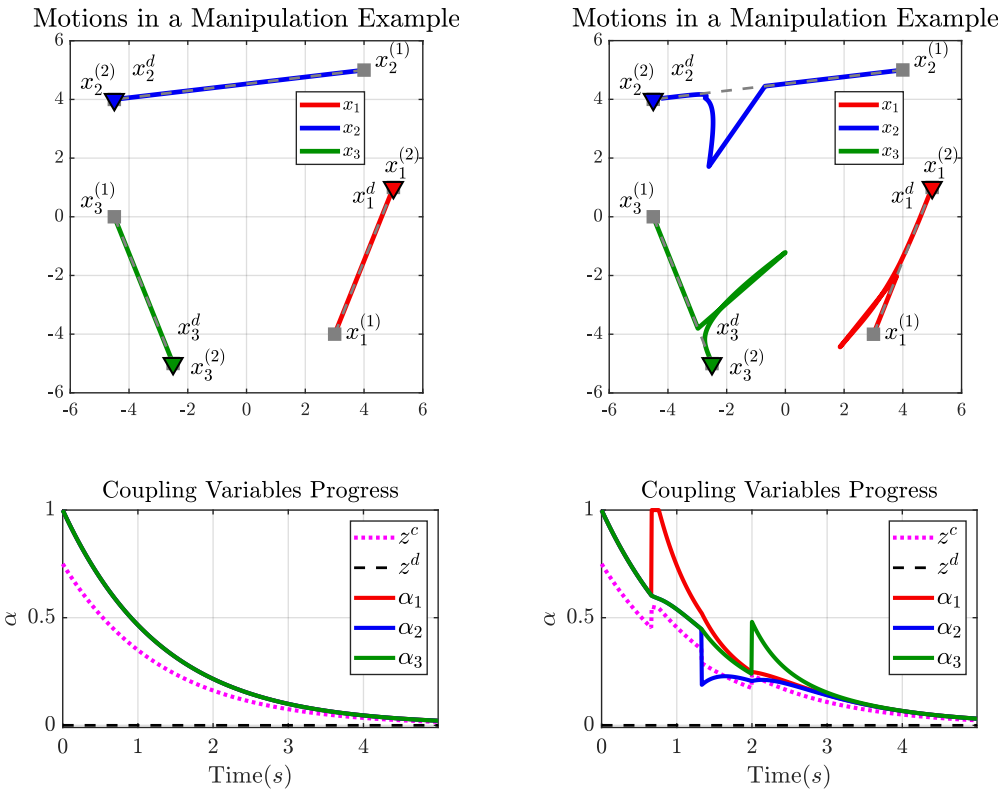


Figure 5.4: An example of planning for manipulation where an object is first grasped in positions $\boldsymbol{x}_1^{(1)}$, $\boldsymbol{x}_2^{(1)}$, and $\boldsymbol{x}_3^{(1)}$. Then the fingers are tasked to rotate the object within the hand frame by simultaneously moving to $\boldsymbol{x}_1^{(2)}$, $\boldsymbol{x}_2^{(2)}$, and $\boldsymbol{x}_3^{(3)}$. Figures on the left show the ideal case, and figures on the right depict the same examples where the fingers react to perturbation at different times.

points is the core to our planning approach for grasp (see Section 5.5.1) and manipulation (see Section 5.5.2). $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ set the positions, respectively, where a task begins and terminates. For instance, consider an object manipulated by three fingers. The task is to rotate the object on a plane, see Figure 5.4. In such a case, fingers are required to move synchronously throughout the entire trajectory. For all fingers, $\{\boldsymbol{x}_i^{(1)}\}_{i=1}^{n_f}$ is the initial fingertip configuration, and $\{\boldsymbol{x}_i^{(2)}\}_{i=1}^{n_f}$ is the final one, after a full rotation.

Robustness of DS

A key concern in our approach, which we assess in several parts, is robustness to uncertainties and perturbations. Our coupling variable (Eq. 5.10) is formulated in such a way that, if a finger is perturbed, the other fingers will wait instead of retracting, see Figure 5.5. This feature of coupling is critical for grasp and manipulation, as a perturbation on one finger should not cause other fingers to break contact and leave the object's surface. We also show that reaction to perturbation is not arbitrary and that a fingertip will restore the intended trajectory (Figure 5.3), which is critical in grasp and manipulation.

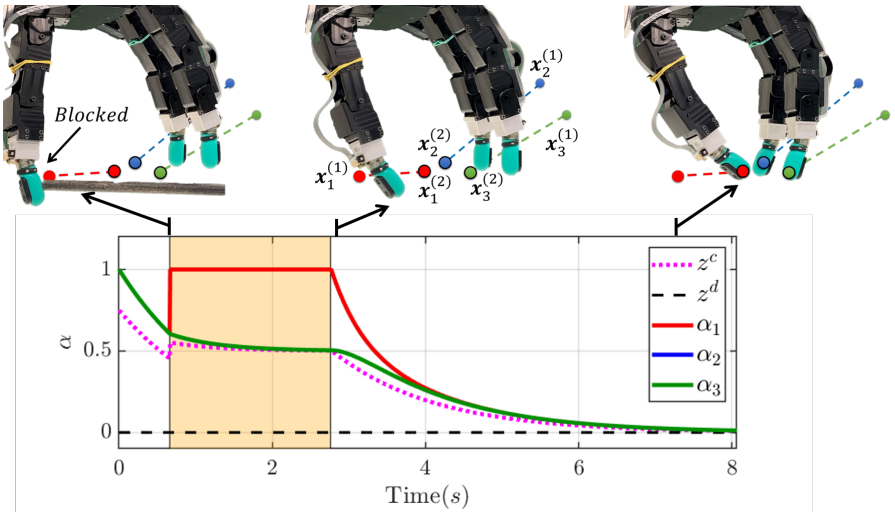


Figure 5.5: Response of the coupled multi-fingered system to a disturbance in grasp planning. Whereas the first fingertip (thumb) is blocked for a couple of seconds, the other fingers wait (the shaded area) for the thumb to rejoin them.

5.4 Joint-Space Adaptive Controller

5.4.1 Low-Level Control

We estimate online the function $u_i(\cdot)$ in Eq. 5.2. The objective of control law $u_i(\cdot)$ is to compute a set of joint torques $\{\boldsymbol{\tau}_i\}_{i=1}^{n_f}$ that realizes the desired fingertip velocities $\{\dot{\boldsymbol{x}}_i^d\}_{i=1}^{n_f}$ given by DSs (5.9). As the control laws for all fingers are derived in a similar fashion, for better readability, we drop the finger-dependent indices i for each i-th finger in this section.

We perform control and adaptation in the joint space, as each finger joint differs in its mechanical imperfections and physical properties hence requires different adaptation laws. We propose a group of adaptive laws for learning and controlling the system's dynamics under uncertainties. We first introduce a nominal dynamics to be the reference model. Then, the controller is tasked to ensure that the actual dynamics of joints perfectly follows this reference model.

5.4.2 Nominal Joint-Space Dynamics

Let's express the controllable states of the system, $\boldsymbol{\zeta} = [\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T]^T \in \mathbb{R}^{2n_j}$, consisting of the joint position, $\boldsymbol{q} \in \mathbb{R}^{n_j}$, and joint velocity, $\dot{\boldsymbol{q}} \in \mathbb{R}^{n_j}$ for each i-th finger with n_j number of joints. $\boldsymbol{\zeta}$ is needed to follow a reference vector $\boldsymbol{\zeta}_r$ that evolves based on a nominal dynamics of the form:

$$\dot{\boldsymbol{\zeta}}_r = \mathbf{A}_r \boldsymbol{\zeta}_r + \mathbf{B}_r \boldsymbol{r}(\dot{\boldsymbol{x}}^d), \tag{5.15}$$

in which $\mathbf{A}_r \in \mathbb{R}^{2n_j \times 2n_j} \succ 0$, and $\mathbf{B}_r \in \mathbb{R}^{2n_j \times n_j}$, $rank(\mathbf{B}_r) = n_j$, are designed to shape the reference model. They are set such that Eq. 5.15 is asymptotically stable at a desired state $\boldsymbol{\zeta}^d$. Matrix $\mathbf{A}_r$ controls how fast $\boldsymbol{\zeta}_r$ converges to $\boldsymbol{\zeta}_i^d$ while $\mathbf{B}_r$ regulates $\boldsymbol{\zeta}_r$ to track $\boldsymbol{\zeta}^d$. Moreover, $\boldsymbol{r}(\dot{\boldsymbol{x}}^d) \in \mathbb{R}^{n_j}$ is a bounded regulation signal.

Remark 2 *$\boldsymbol{r}(\dot{\boldsymbol{x}}^d)$ links the desired fingertip velocity $\dot{\boldsymbol{x}}^d$ (Eq. 5.1 and Eq. 5.9) to the joint-space desired state $\boldsymbol{\zeta}^d$. Appendix B.1.1 provides further details on how to compute $\boldsymbol{r}(\dot{\boldsymbol{x}}^d)$ from $\dot{\boldsymbol{x}}^d$.*

For brevity, henceforth we drop the dependency of $\boldsymbol{r}$. The nominal dynamics takes the desired velocity from DS (Eq. 5.9) as input; however, note that the concept of nominal dynamics and actual dynamics differ from DSs explained in task planning, *Section 5.3*. Here, we shape the dynamic behavior of each joint, regardless of the desired task (target) at hand. As a result, the controller that we develop in this section can be used with other planning approaches as well.

5.4.3 Control Rule and Adaptive Laws

The governing dynamics of each finger can be represented by a DS (Culbertson and Schwager (2018)) of the form:

$$\dot{\boldsymbol{\zeta}} = \mathbf{A}\boldsymbol{\zeta} + \mathbf{B}\boldsymbol{\nu}. \tag{5.16}$$

The control effort $\boldsymbol{\nu}$ corresponds to the joint torques of the i-th finger ($\boldsymbol{\tau}_i$ in Eq. 5.2). Let's assume that $\mathbf{A} \in \mathbb{R}^{2n_j \times 2n_j}$, and $\mathbf{B} \in \mathbb{R}^{2n_j \times n_j}$ are unknown and can vary with time as the evolution of the fingers' states is subjected to uncertainties.

The controller's objective is to ensure that $\boldsymbol{\zeta}$ tracks the reference dynamics model (Eq. 5.15) in presence of nonlinearities and uncertainties. To achieve this, we propose an adaptive mechanism to modulate ν:

$$\boldsymbol{\nu} = \Psi_\zeta \boldsymbol{\zeta} + \Psi_\mathrm{r} \boldsymbol{r}. \tag{5.17}$$

The adaptive control gains $\Psi_\zeta \in \mathbb{R}^{n_j \times 2n_j}$, and $\Psi_\mathrm{r} \in \mathbb{R}^{n_j \times n_j}$ are estimated/trained online to compensate for changes in the dynamics of the fingers and the tracking error $\boldsymbol{e} = \boldsymbol{\zeta} - \boldsymbol{\zeta}_i^d$, using the following update rules:

$$\begin{cases} \dot{\bar{\Psi}}_\zeta &= -\Lambda_\zeta \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{\zeta}^T - \mathbf{S} \Upsilon_\zeta \bar{\Psi}_\zeta \\ \dot{\bar{\Psi}}_\mathrm{r} &= -\Lambda_r \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{r}^T(t) - \mathbf{S} \Upsilon_\mathrm{r} \bar{\Psi}_\mathrm{r} \end{cases} \tag{5.18}$$

wherein:

$$\mathbf{S}_{jj} = \max\big(0, \boldsymbol{\epsilon}_j - e^{-\frac{e_j^2}{2\sigma^2}}\big)$$
$$\Upsilon_{\boldsymbol{\zeta}\,jj} = \boldsymbol{\kappa}_j^* \big(1 - \frac{1}{(\bar{\Psi}_{\boldsymbol{\zeta}} \bar{\Psi}_{\boldsymbol{\zeta}}^T)_{jj} + \iota}\big)$$
$$\Upsilon_{\mathrm{r}\,jj} = \boldsymbol{\kappa}_j^* \big(1 - \frac{1}{(\bar{\Psi}_{\mathrm{r}} \bar{\Psi}_{\mathrm{r}}^T)_{jj} + \iota}\big)$$

where $\Lambda_\zeta \in \mathbb{R}^{n_j \times n_j}$, and $\Lambda_r \in \mathbb{R}^{n_j \times n_j}$ are positive definite matrices that tune convergence rate of the adaptive gains, whereas $\mathbf{P} \in \mathbb{R}^{2n_j \times 2n_j} \succ 0$ forms a quadratic Lyapunov function, see Appendix 2.1. $\boldsymbol{\kappa}_j^*$ and $\boldsymbol{\epsilon}_j$ are the desired stiffness and the allowed error-tolerance for j-th joint of each finger. $\boldsymbol{\kappa}_j^*$ and $\boldsymbol{\epsilon}_j$ are fixed values defined by the user (we set $\boldsymbol{\kappa}^* = [0.15, 0.12, 0.1, 0.05]$ for each finger, and $\boldsymbol{\epsilon}_j = 0.02$ for all joints in our experiments), and they avoid saturation in control signal $\boldsymbol{\nu}$, as well as in the adaptive gains Ψ_ζ, and Ψ_r.

Theorem 4 *Using the control law given by Eq. 5.17 and the adaptive laws (Eq. 5.18), the system (5.16) asymptotically converges to the nominal dynamics (5.15), and* $\lim_{t\to\infty} \|\boldsymbol{\zeta}^d - \boldsymbol{\zeta}\| \to 0$ *if there exist* $\mathbf{P} > 0$ *and* $\mathbf{Q} > 0$ *such that* $\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$

Proof *See Appendix B.1.2*

The diagonal matrix $\mathbf{S}$ acts as an activation function for stiffness regulation inside the error bound $\boldsymbol{\epsilon}$; for instance, setting $\boldsymbol{\epsilon}_j = 0$ deactivates this feature of the controller for the j-th joint of the finger. Appendix 2.1 provides insight into how to design $\mathbf{P}$, $\mathbf{A}_r$, and $\mathbf{B}_r$.

Note that in the first time-step, we initialize adaptive gains, Ψ_ζ and Ψ_r, with a guessed value. Then, at the end of each iteration, we update their values via adaptive law in Eq. 5.18. The adaptation rate has to be faster than the actual dynamics. Once a task is completed, we can store the trained Ψ_ζ, and Ψ_r, and use them as the initial guess for new trials. Algorithm 3 summarizes our control framework that will be later used in grasp and manipulation tasks.

5.5 Grasp and Manipulation

Thus far, we have formulated a robust, stable, and finger-synchronized controller for a robotic hand. Here, we discuss in detail how this control framework is used for grasping and manipulating objects; see Figure 5.6 for an illustration. First, we optimize contact wrenches to stabilize the object in the current configuration. Then using the the object-level impedance, we find a set of attractors $\boldsymbol{x}_i^{(1)}$ and $\boldsymbol{x}_i^{(2)}$ for each i-th finger, see Figure 5.4. These attractors are coordinated in the grasping process in order to insert force to the object in a synchronized fashion.

Algorithm 3 Coordinated adaptive torque control for each i-th finger.

Input:
  $\boldsymbol{x}_i, \dot{\boldsymbol{q}}_i, \boldsymbol{q}_i, \boldsymbol{\eta}_i, \boldsymbol{x}_i^{(2)}$
Output:
  $\boldsymbol{\tau}_i$
Parameters:
  β_1, β_2 (Eq. 5.5), and Λ_ζ, Λ_r (Eq. 5.18)
Defining DS Models:
  $\mathbf{A}_x \succ 0$ for DS (Eq. 5.9)
  $\mathbf{A}_r \succ 0, \mathbf{B}_r\ (rank(\mathbf{B}_r) = n_j)$
  $\mathbf{P} \succ 0$ that $\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$ for DS (Eq. 5.15)
Initialize:
  $\bar{\Psi}_{\boldsymbol{\zeta}}, \bar{\Psi}_r$
for $t \leftarrow 0 \rightarrow T$ **do**
  $\boldsymbol{x}_i, \dot{\boldsymbol{q}}_i, \boldsymbol{q}_i \leftarrow$ robot's feedback
  $\alpha_i \leftarrow$ Eq. 5.7 $\leftarrow \boldsymbol{x}_i$
  $z^c \leftarrow$ Eq. 5.10 $\leftarrow \{\alpha_i\}_{i=1}^{n_f}, z^d$
  $\dot{z}_i \leftarrow$ Eq. 5.5 $\leftarrow z^c, \alpha_i$
  $z_i \leftarrow \max\big(0, \min(1, \dot{z}_i\delta + \alpha_i)\big)$
  $\dot{\boldsymbol{x}}_i^d \leftarrow$ Eq. 5.6
  $\boldsymbol{r}, \boldsymbol{\zeta}_i^d \leftarrow$ Eq. B.1
  $\boldsymbol{\zeta} \leftarrow [\dot{\boldsymbol{q}}_i^T, \boldsymbol{q}_i^T]^T$
  compute $\boldsymbol{\tau}_i$ from Eq. 5.17
  update $\bar{\Psi}_{\boldsymbol{\zeta}}, \bar{\Psi}_r$ by Eq. 5.18
end for

5.5.1 Contact Wrench Optimization

Grasp Configuration

When grasping an object, we can heuristically estimate a stable grasp configuration, i.e., the combination of finger placements on the object surface. This estimation might be sub-optimal compared to solutions in interesting studies such as (Li et al. (2016a); Dragiev et al. (2011); Hang et al. (2020)). However, an optimal grasp configuration is never stable if contact forces are erroneously applied (Roa and Suárez (2015)). In our experiments, grasp configuration is heuristically set (the thumb opposes the index and middle finger). Then, we focus on force adaptation at contact points, whether or not a grasp configuration is optimal.

Optimization

Once the finger-object contacts are established, each finger is needed to apply a contact force, $\lambda_i \in \mathbb{R}^3$, to stabilize the object. To do so, we adopt the force closure linear programming-based optimization in (Prattichizzo and Trinkle (2016)), determining the desired contact wrenches $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1^T, ..., \boldsymbol{\lambda}_{n_f}^T]^T$:

$$\begin{aligned} \text{maximize} \quad & \eta \\ \text{s.t.} \quad & \mathbf{G}\boldsymbol{\lambda} = 0 \\ & \mathbf{F}\boldsymbol{\lambda} - \mathbf{1}_n \eta \geq 0 \\ & \eta \geq 0 \\ & \boldsymbol{e}\boldsymbol{\lambda} \leq \boldsymbol{n}_f \end{aligned} \tag{5.19}$$

$\eta \in \mathbb{R}^+$ is the optimization variable. $\boldsymbol{e} = [\boldsymbol{e}_1, ..., \boldsymbol{e}_{n_f}]$, and $\boldsymbol{e}_i = [1,0,0]$. $\mathbf{F} = diag(\mathbf{F}_1, ..., \mathbf{F}_{n_f})$ is the friction cone matrix, and $\mathbf{G}$ represents the grasp matrix where $\mathbf{F}_i \in \mathbb{R}^{3\times3}$ and $\mathbf{G}_i \in \mathbb{R}^{3\times3}$ are, respectively, the block of $\mathbf{F}$ and $\mathbf{G}$ corresponding to the i-th contact point. The grasp matrix maps contact wrenches to the object twist, and it is computed given contact frames, contact properties, and object's center of mass, follow (Prattichizzo and Trinkle (2016)).

For each contact point, the friction law puts constraints on the contact wrench $\boldsymbol{\lambda}_i$, defined by the type of contact (soft/hard finger) and by the friction coefficient μ. Here we consider the hard finger (HF) contact type (no twist at contact point) and the friction to be $\mu = \mu_n - \tan(e_{nn})$ where μ_n is the nominal friction and e_{nn} is the absolute error in contact normal ($\hat{\boldsymbol{n}}_i$) estimation discussed in *Section 5.5.1* and provided in Table B.1.

Contact-Frame Estimation

In optimization (5.19), each contact wrench is expressed in a frame $\{C_i\} = \{\hat{\boldsymbol{n}}_i, \hat{\boldsymbol{t}}_i, \hat{\boldsymbol{o}}_i\}$, which encapsulates the object-shape information at i-th contact point. The unit vector $\hat{\boldsymbol{n}}_i \in \mathbb{R}^3$ is the contact normal, directed toward the object. The other two vectors are orthogonal and lie in the tangent plane of the contact. Moreover, the computation of grasp matrix $\mathbf{G}$ and friction cone matrix $\mathbf{F}$ are based on the contact frames and their positions. We use tactile observation to estimate vector $\hat{\boldsymbol{n}}_i$ of each contact. In essence, we use a mapping from electrodes impedance $\boldsymbol{\rho}_i \in \mathbb{R}^{19}$ of tactile sensor to vector $\hat{\boldsymbol{n}}_i$ at the contact point: $\hat{\boldsymbol{n}}_i = f_{ANN}(\boldsymbol{\rho}_i)$, see Appendix B.2 for details on how the mapping is learned.

5.5.2 Attractors Determination

Grasp

With contact wrenches from optimization (5.19), we set the attractors of the DS (5.9) as

$$\boldsymbol{x}_i^{(2)} = \boldsymbol{K}_x^{-1} \mathbf{G}_i \boldsymbol{\lambda}_i + \boldsymbol{x}_i^{(1)} \tag{5.20}$$

$\boldsymbol{x}_i^{(1)}$ is the contact position on the object surface. $\boldsymbol{K}_x$ is the stiffness coefficient chosen according to the desired finger-object compliance. Therefore, given the analysis in Section 5.3.3, applying the wrench force will be in a synchronized fashion, which will guarantee a robust grasp within the fingers.

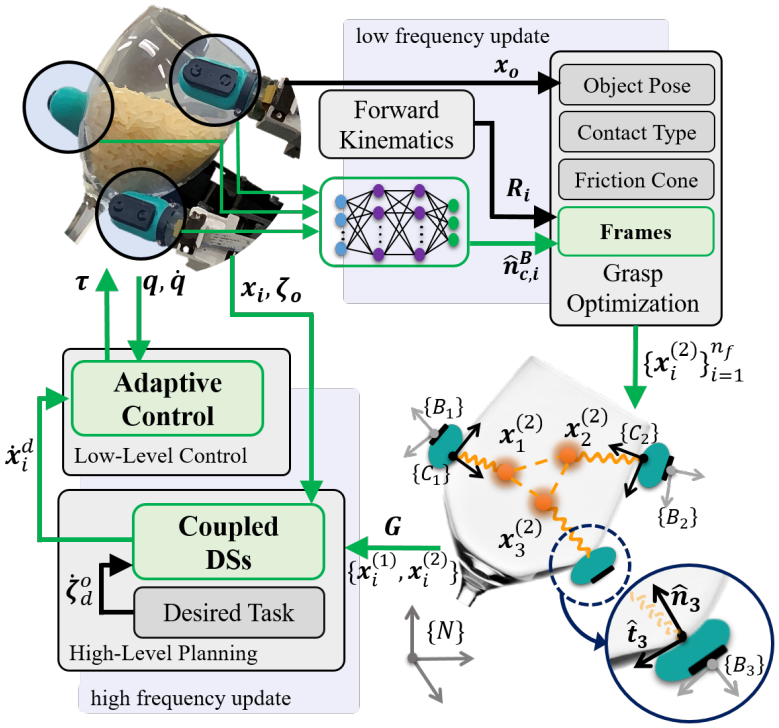


Figure 5.6: Block diagram of our method for grasp and manipulation using a coordinated adaptive torque-control. Contact frames are estimated with 60Hz frequency by using tactile sensors and the learned neural-network model. These frames are then sent to the grasp optimization algorithm to construct an appropriate object-level impedance (orange elements). Finally, the extracted attractors are fed to the devised controller that runs with 200Hz frequency. Components with a green color signify the focus of this study and our contribution to the state of the art.

In-Hand Manipulation

Once the object is securely grasped, in-hand manipulation is generated by re-positioning the set of attractors $\{\boldsymbol{x}_i^{(2)}\}_{i=1}^{i=n_f}$. Attractor re-positioning for DS (5.9) is done using the grasp matrix for computing the desired object velocity:

$$\boldsymbol{x}_i^{(2)} = \mathbf{G}_i^T \dot{\boldsymbol{\zeta}}_o^d \delta + \boldsymbol{K}_x^{-1} \mathbf{G}_i \boldsymbol{\lambda}_i + \boldsymbol{x}_i^{(1)} \tag{5.21}$$

here $\boldsymbol{x}_i^{(1)}$ is the contact point, similar to Eq. 5.20. $\boldsymbol{\zeta}_o \in \mathbb{R}^{d_x}$ represents the states of the object; for instance in a planar motion, $\boldsymbol{\zeta}_o = [x_o, y_o, \theta_z]$. Note that the process of defining attractors in manipulation is compatible with grasp, hence the system will remain coupled throughout the entire procedure of first grasping and then manipulating the object. In our control approach, the fingers are dynamically coupled and the manipulation will be executed by all fingers synchronously contributing to the task (unless a finger has been exempted or has a different role, see *Section 5.6.4*). Therefore, similar to grasping, the coupling DS (Eq. 5.5) has to be computed and updated online.

5.6 Experiments and Evaluations

We evaluate our approach qualitatively and quantitatively in four experiments with a real robotic hand, Allegro V4, mounted on a robotic arm, KUKA LBR iiwa 14. We perform the following assessments:

(i) We evaluate the quality of control, adaptation, and coordination of fingers, in *Section 5.6.1*.

(ii) In *Section 5.6.2*, we evaluate the robustness of the grasped object with different shapes and mass properties Figure 5.9. This experiment is based on the approach summarized in Figure 5.6 and we study grasp adapting in an uncertain environment.

(iii) We perform in-hand manipulation (translation and rotation) of four objects of different shapes, and mass properties, in *Section 5.6.3*.

(iv) In *Section 5.6.4*, we augment our control approach with learning from demonstration to execute a complex manipulation task: in-hand object rolling. We devise this experiment to give an instance of a dynamic disturbance during in-hand manipulation.

5.6.1 Evaluation of Coordinated Finger-Control

We test the strength of our coordinated fingers control approach in two scenarios:

(i) We fix the base of Allegro hand in a specific pose. The fingertips need to first move from *rest* positions to *ready* positions and, then, to track a specific sequence of attractors, see Figure 5.7.

(ii) We compare our control algorithm against an impedance-based controller with a state-varying impedance called “Passive-DS" (Kronander and Billard (2015)). The task here is similar to the first scenario, except that it is repeated with the base of the hand being fixed in three different poses.

Tracking Performance and Dynamic Learning

In this experiment, all adaptive gains Ψ_ζ and Ψ_r for all fingers are initiated with zero value matrices and $\boldsymbol{\kappa}^* = [0.15, 0.12, 0.1, 0.05]$ for each finger and $\boldsymbol{\epsilon}_j = 0.02$ for all joints. Figure 5.7 shows the performance of our controller in the first experiment where the tracking error decreases simultaneously for all fingers. We observe, in displacement from rest positions to ready positions, higher oscillations in error compared to the transitions in other task segments (for instance from $\{\boldsymbol{x}_i^{(1)}\}_{i=1}^{i=n_f}$ to $\{\boldsymbol{x}_i^{(2)}\}_{i=1}^{i=n_f}$). In the first part, adaptive gains are initialized as zero matrices and the oscillation is due to controller exploration to find the proper range of gains. Whereas, in the other parts, gains are only fine-tuned in that specific range, see

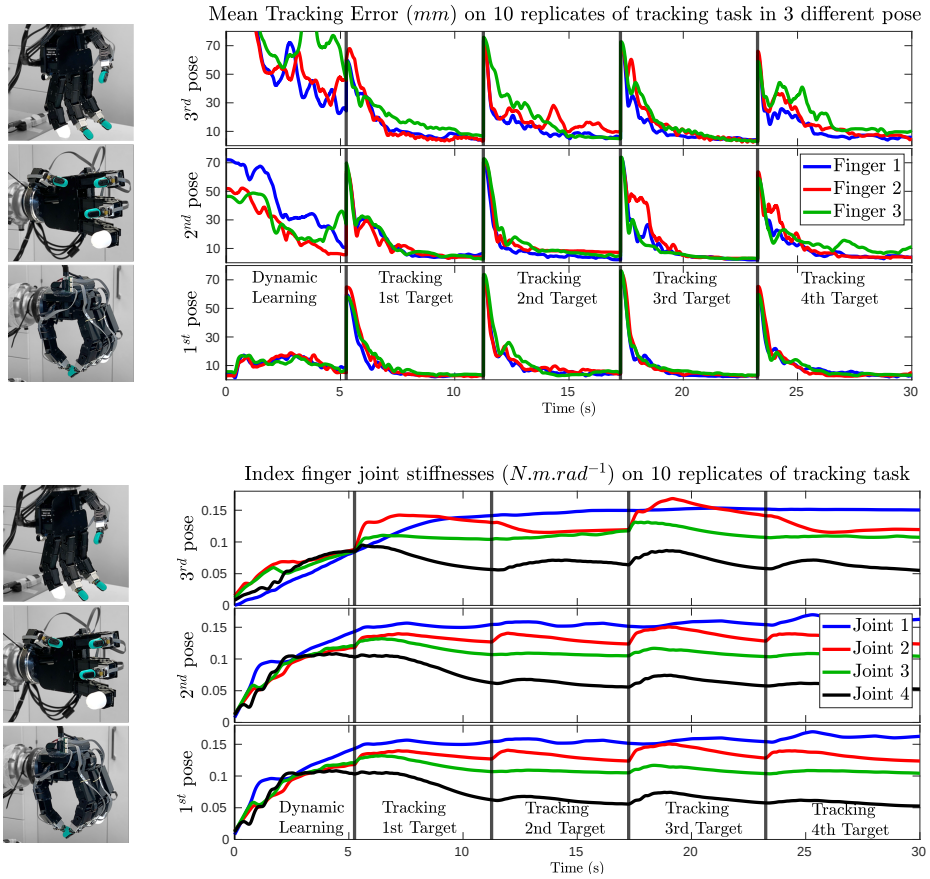


Figure 5.7: Evaluation of the coordinated adaptive torque control on a robotic hand in three different wrist poses (different gravity and dynamics effect). In the first phase of the task, the fingers move from "rest" positions to "ready" positions, wherein first the unknown DS (Eq. 5.16) is learned with adaptive gains converging. Then, the task is to track a specific sequence of different target positions. From left to right: (left): Mean tracking error (mm) over 10 replicates, (middle): robotic hand in three different poses, and (right): index finger joints' stiffness ($\boldsymbol{\kappa}$ in Eq. 5.18) which are required to converge to the desired values, $\boldsymbol{\kappa}^* = [0.15, 0.12, 0.1, 0.05]$.

the right figure in Figure 5.7. Furthermore, using coupled dynamical system when moving from $\{\boldsymbol{x}_i^{(m)}\}_{i=1}^{i=n_f}$ to $\{\boldsymbol{x}_i^{(m+1)}\}_{i=1}^{i=n_f}$ results in synchronous convergence with lower oscillation in tracking. While converging to their attractors, all fingers will simultaneously regulate the joints' stiffness to desired values $\{\boldsymbol{\kappa}_j\}_{j=1}^{j=n_j}$. The regulation of joint impedance not only enables us to better control finger-object interactions later but also avoids joint saturation due to large initial errors, thus resulting in higher adaptation efficiency.

Comparison with Impedance-Based Controllers

The purpose of this experiment is to compare the performance of our low-level adaptive torque controller with impedance-based controllers as the baseline. Considering related

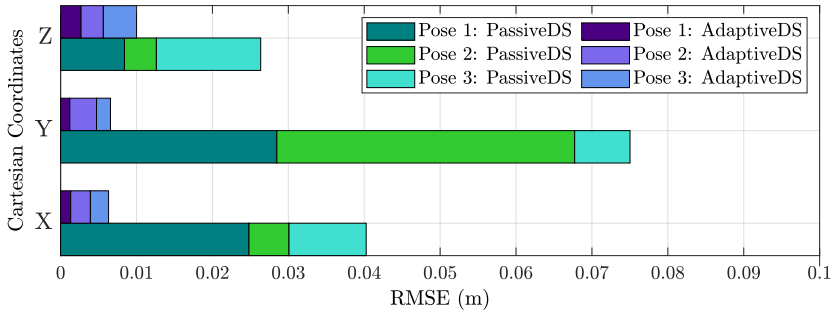


Figure 5.8: Tracking RMSE of each Cartesian coordinate for our adaptive controller and a passive-DS with large gains. This experiment is similar to the one in Figure 5.7 and it is replicated 20 times for each specific pose.

studies with real robot evaluation, we find that impedance-based controllers are primarily employed to control the fingertip positions of the robotic hand (Sundaralingam and Hermans (2019); Li et al. (2014, 2016a)). We compare our method with the passive-DS control used in (Li et al. (2016a)) for grasping. Passive-DS control (Kronander and Billard (2015)) is one variation of impedance-based controllers, and it has an improved structure compared to the PD control employed in (Sundaralingam and Hermans (2019)). Passive-DS is compatible with the dynamical systems. Thus by using this controller, we constrain the experiment such that the only feature to be tested is the performance of the low-level controller.

We fix the base of the hand in three different poses. Then, we perform separately a tracking task by using two controllers: (a) our adaptive control algorithm, and (b) a PD controller with a state varying impedance. In the first, second, and third pose the gravity force is parallel to X, Y, and Z coordinates of the hand frame, respectively. The consistency of control performance, given different hand poses, is the key requirement to obtaining a robust and secure grasp. Here, we showcase the strength of our controller for situations where the pose of the hand is not fixed.

After implementing the task using the same control parameters for all poses, we recorded average steady-state errors over 20 replicates for each pose. To make a fair comparison, we pushed the passive-DS control gains to maximum achievable performance, without destabilizing the robot's joints. Figure 5.8 illustrates the results of the second scenario. In all three hand base poses, our controller consistently exhibits a higher accuracy compared to passive-DS (even with large PD gains). This is because we establish our adaptation at joint level to adapt for individual joint uncertainties and specifications, which is a key factor in object manipulation by a robotic hand. When an object is grasped, the uncertainty level and the gravity effect are considerably higher; as expected, our low-level adaptive controller significantly outperforms the impedance-based controllers in such cases.

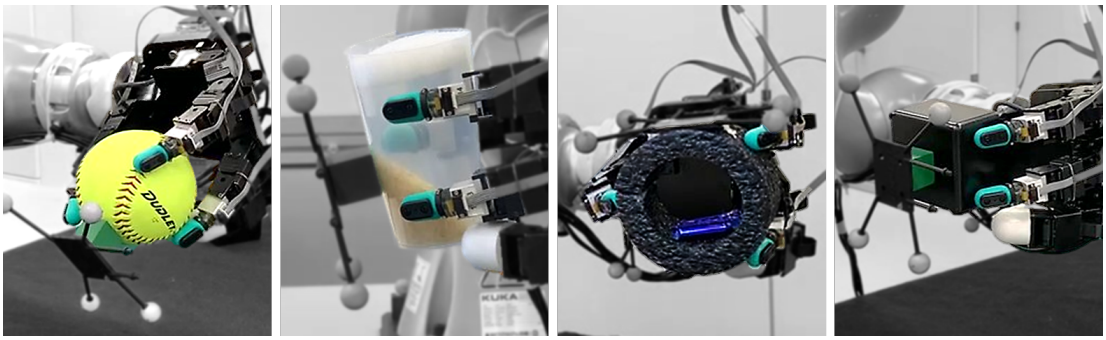


Figure 5.9: Objects used in both grasp and manipulation experiments. From left to right: (1) Baseball (heavy object), (2) Tumbler half full of rice (varying mass distribution), (3) Annulus ring with a mass attached to inner part (uneven mass distribution), and (4) cuboid (normal object, easy to grasp).

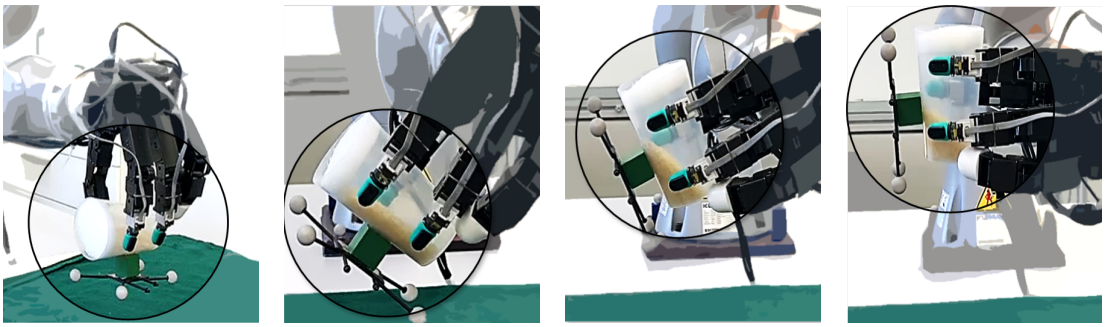


Figure 5.10: Adaptation in grasp experiment when rotating a tumbler half-filled with rice. In this particular case, the mass distribution changes, thus imposing disturbances on fingers stability. Object relative displacement to the hand is recorded via OptiTrack motion capture system.

5.6.2 Grasp Adaptation in an Uncertain Environment

In this part, our focus is on evaluating the grasp robustness for different objects of different properties, see Figure 5.9. In other words, we assess the performance of our controller in adapting the grasp where the robotic hand is needed to displace the object, e.g., hand-over task, and pick and place task. To do so, we devise the following scenarios: Using the same control framework, Figure 5.6, the Allegro hand first grasps the object in a predefined grasp configuration. While the object is held within the hand, the wrist (KUKA's last joint) rotates 90^o clockwise, thereby manipulating the position and the orientation of the object. This task is executed for ten replicates per object. Figure 5.10 depicts an example of it where the tumbler is half-filled with rice (object with varying mass distribution); see also supplementary materials for the videos of the experiment.

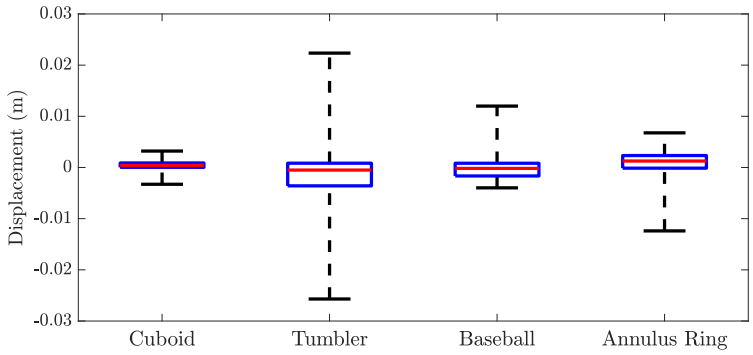


Figure 5.11: Results of the adaptive grasp robustness experiment and rotation experiment replicated ten times (see Figure 5.10 as one example). Box-plots indicate objects relative displacement with respect to the hand's frame during wrist rotation.

Table 5.1: The number of trials to have 10 successful execution for the adaptive grasp robustness experiment.

Object	**Cuboid**	**Tumbler**	**Baseball**	**Annulus Ring**
Trials	10	14	12	12

Results on Grasp Adaptation during Wrist Rotation

In this experiment, the controller has to adapt joint torques to face not only the changes in the gravity direction but also the variation of object inertia. During this task, using OptiTrack motion capture system, we recorded the motion of object's center of mass relative to the hand frame. Figure 5.11 shows the boxplot of ten replicates of the experiment for each object. Given this figure, and Table 5.1, our grasp algorithm is able to hold the object stable with very low relative displacement, and it provides proper robustness despite changes in the inertia of the objects. Slippage, partial occultation of the object from the OptiTrack system in the hand frame (jerk in position feedback), and rapid changes in inertia (faster than the adaptation rate in tumbler case) were the main causes of failed grasps in Table 5.1. In the previous section, we show that our adaptive controller is capable of facing uncertainties at the joint level. Here, as long as the finger joints sense the perturbation from object uncertainties (i.e., fingertips are in contact with objects), our controller is capable of adapting to these uncertainties and remains robust enough to hold the object stable.

Force and Contact Normal-Vector Adaptation

Figure 5.12 depicts an example of contact-force adaptation. For each finger in contact, the vector normal to object surface $\hat{\boldsymbol{n}}_i$ is re-estimated through the trained NN model (Eq. B.6). These estimations are used to improve our prior knowledge of the object's shape, especially at contact points under the fingertips and tactile sensors. $\hat{\boldsymbol{n}}_i$ of fingers in contact is then fed to the grasp optimization process, Figure 5.6, where via shaping the object-level impedance, the

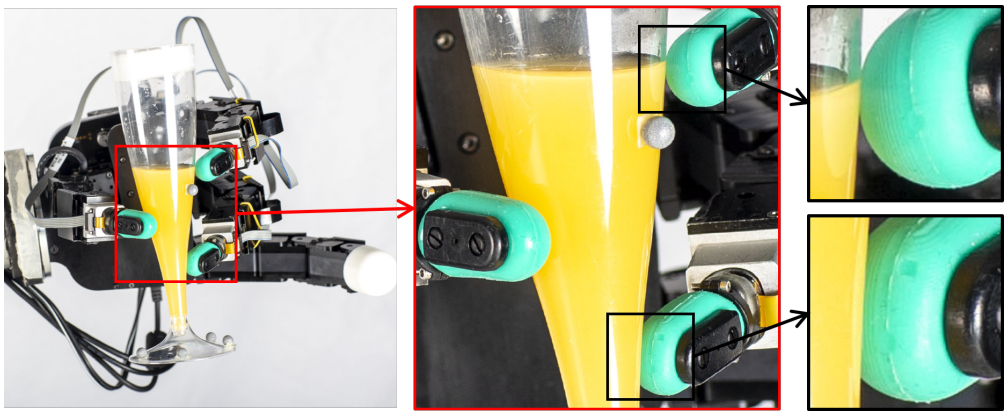


Figure 5.12: An example of contact force adaptation. From the prior knowledge the object is assumed to be of a cylinder shape. Hence fingers arrive at contact each with a different orientation. Given the tactile sensor and the trained NN model (Eq. B.6), the normal vector $\hat{\boldsymbol{n}}_i$ at contact point is estimated for each i-th finger. Approximated contact frames are then fed to grasp optimization process (5.19) in order to extract contact wrenches.

force direction for each fingertip is extracted. In Figure 5.12, we observe that the robotic finger updates the force direction (pushing more toward the center of the ring and increasing the contact area), thereby securing grasp stability.

5.6.3 In-Hand Manipulation, Accuracy and Robustness

The experiments in this section are devised to manipulate the objects listed in Figure 5.9. Assuming that the objects are grasped perfectly, in order to realized a desired pose, the controller is needed to apply force to the object through the fingers by using Eq. 5.21 and Eq. 5.9; see Figure 5.13 as an example and see supplementary videos for the rest of objects. For comparison purposes, the desired pose manipulations are selected such that within the workspace of the hand, they are feasible for all objects. We also use in-hand manipulation in a gearbox assembly task (presented in supplementary videos) as an application of in-hand manipulation.

Results

Table 5.2 summarizes the results of ten replicates of a specific manipulation, which reveals a high tracking accuracy of our control approach. Failed attempts in Table 5.2 similar to Table 5.1 were mainly due to slippage, or partial occultation of the object from OptiTrack system. Figure 5.14 presents the tracking error profiles of 30^o rotation for all objects of the experiment. Apart from accurate tracking, one observation from Figure 5.14 is that, unlike grasping, objects with more regular shape or mass are not necessarily easier to manipulate. For instance, the mean tracking error for the cuboid is higher than the other objects in Figure 5.14, but it has the highest stable grasp (Table 5.1 and Figure 5.11). This is otherwise unexpected if we do not consider the importance of robotic hand's dynamics.

Table 5.2: Summary of results on three in-hand manipulations tasks (20mm of translation, 30^o, and 45^o of rotation) on four objects in Figure 5.9. "Attempts" numbers indicate the number of trials to have 10 successful execution, and "Error" values shows the steady-state tracking error of successful trials.

20 mm				
Objects	**Cuboid**	**Tumbler**	**Baseball**	**Ring**
Error (%)	-1±10	.2±10	.4±13	.1±11
Attempts	10	11	11	10
30^o				
Objects	**Cuboid**	**Tumbler**	**Baseball**	**Ring**
Error (%)	8.7±3.8	-.5±5.5	-.3±11	-1.3±4.7
Attempts	11	12	11	10
45^o				
Objects	**Cuboid**	**Tumbler**	**Baseball**	**Ring**
Error (%)	5.6±4.0	-.5±5.4	-1.0±5.0	.7±4.9
Attempts	12	13	14	11

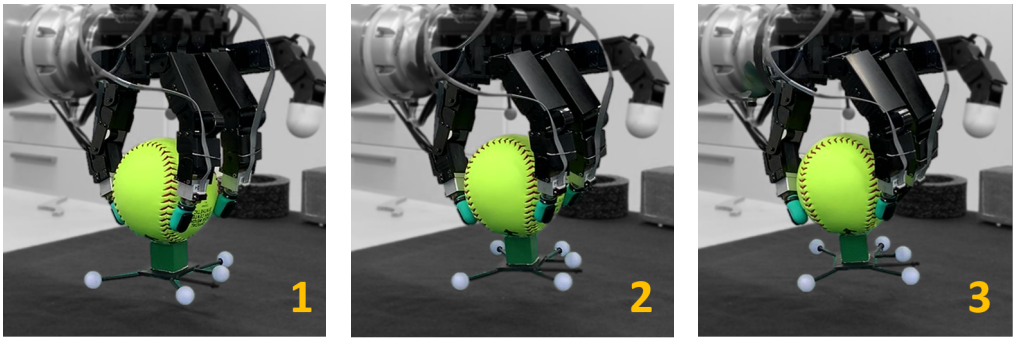


Figure 5.13: A baseball, as an example of a heavy object, is rotated 30^o. The task is performed using DS (Eq. 5.9) and the control approach in Figure 5.6. The mean rotation error for 10 replicates of this experiment is $-2 \times 10^{-3} rad$, see Table 5.2.

Comparing with the State of the Art

Using Allegro hand in a similar experiment, Sundaralingam and Hermans (2019) performed multiple in-hand manipulations on 10 objects from the YCB dataset. To move an object, they regulated the impedance of the grasp in the direction of the objective pose via kinematic trajectory optimization. Comparing Table 5.2 with the reported manipulation results in (Sundaralingam and Hermans (2019)), we observe that the highest margin of position error (13% for the *baseball*) in Table 5.2 is smaller than the lowest reported median error for an object ($\approx$ 20% for the *Lego* object) in (Sundaralingam and Hermans (2019)). Similarly for the orientation, the reported median error (9.86%) in (Sundaralingam and Hermans (2019)) is higher than the error margin in Table 5.2. This difference can be explained by the fact that our low-level control adapts to dynamic uncertainties that results in tracking errors, contrary to the PD controller used in (Sundaralingam and Hermans (2019)), see Section 5.6.1.

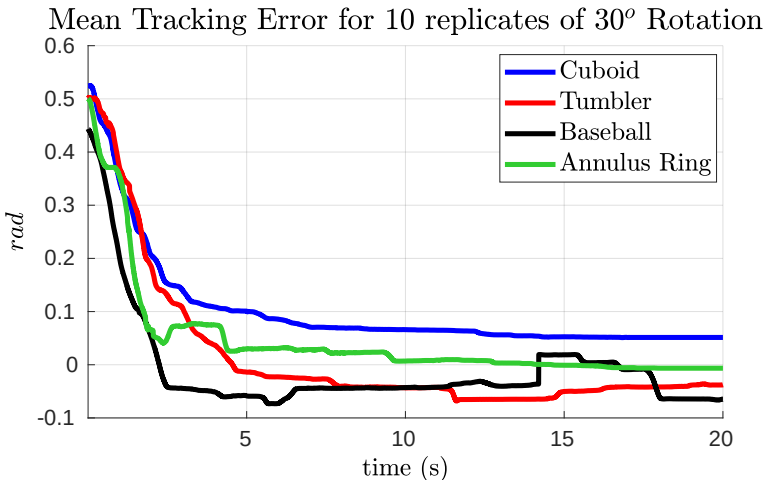


Figure 5.14: Mean tracking error for one of the manipulation experiments. In this task, we rotate different objects listed in Figure 5.9 for 30^o in hand.

5.6.4 Learning to Roll in Hand

In this experiment, we use our control approach to accomplish a complex manipulation task: in-hand object rolling, otherwise known as finger-gaiting. In this task, the hand rotates the object through a combination of contact/non-contact movements, see Figure 5.15 and 5.16. Human demonstrations (see Section 5.6.4) are used to extract the role of each finger and to identify the segments of the task through hidden Markov models (HMM) (Niekum et al. (2012)). We show that task planning in this fashion in conjunction with our control framework enables us to encode a robust solution for in-hand rolling.

Task Segmentation

We employed the sticky hierarchical Dirichlet process (HDP)-HMM in (Fox et al. (2008)) to find the optimum number of states and their sequence. HDP-HMM is applied on the recorded $\mathscr{D}_o = \{{}^o\boldsymbol{\psi}_t^{(i)}\}_{t=0..T_i}^{i=1,n_e}$ of the object's rotations (finger gaiting), where we stored the fingers' relative angular velocities ω and pressures p as observations with ${}^o\boldsymbol{\psi}_t^{(i)} = [\omega_1^T, p_1, \cdots, \omega_4^T, p_4]^T$. The HMM model extracted 3 hidden states each corresponding to a task segment; see Section 5.6.4 for further detail.

Data Collection

We record the object's pose, the finger positions, and contact forces from human demonstrations. The position information from expert demonstrations is recorded at 250 Hz by using the *OptiTrack Motion Capture* system, see Figure 5.17. The contact wrenches are recorded with a set of TPS tactile sensors, thus providing a measure of exerted pressure at 50 Hz, at the fingertips. Next, each demonstration is expressed in sets of data represented in the object frame at

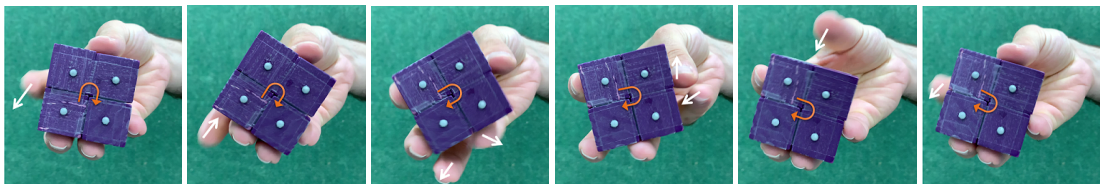


Figure 5.15: From left to right, a human performing finger gaiting in-hand manipulation: the object is rotated clockwise and motion direction are depicted by arrows. The fingers have different roles and contributions to the task. The kinesthetic demonstrations, including object and finger motions, are later used in high-level planning and learning task segmentation through HMM.

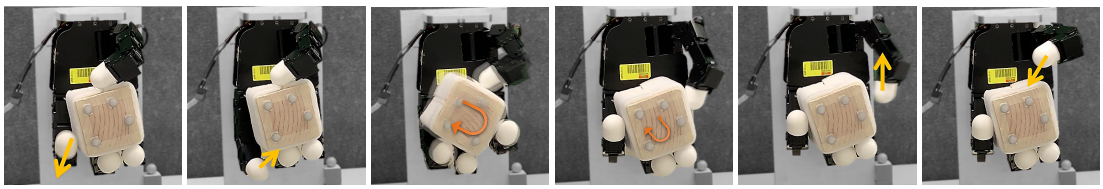


Figure 5.16: From left to right, the robotic hand performs in-hand rolling (finger gaiting) manipulation: task planning and finger roles distributions are learned from human demonstrations. Object-finger interactions in contact and non-contact displacement are torque controlled using our adaptive controller to stably manipulate the object within the hand workspace.

center of mass : $\mathscr{D}_o = \{{}^o\boldsymbol{\psi}_t^{(i)}\}_{t=0..T_i}^{i=1,n_e}$ where ${}^o\boldsymbol{\psi}_t^{(i)} = [{}^o\boldsymbol{x}_1(t)^T, p_1(t), \cdots, {}^o\boldsymbol{x}_{n_f}(t)^T, p_{n_f}(t)]^T$, and n_e is the number of recorded demonstrations. Recorded data $\mathscr{D}_o$ explains how the fingertips move relative to the object. This data is then used to identify the task segments, grasp positions, and the finger roles.

Model Selection

Two probabilistic approaches are prevalent for decomposing the human motions into meaningful actions or states: (i) Gaussian Mixture Models (GMM) (Lee et al. (2015)), where segments are extracted by fitting a GMM to the set of demonstrations, considering each GMM component as one segment, and (ii) Hidden Markov models (HMM) (Niekum et al. (2012)) that use a Markov process to explain the probabilities of sequences of hidden events (states) from possible observations (emissions). The latter is more suitable for the problem at hand, because the segmentation is not only based on a spatial sense but also on the transition dynamics across segments, and because HMM addresses transition dynamics handled through Markovian states. Given the dataset $\mathscr{D}_o$, we independently segment each trajectory while discovering the sequence of observations over T_i discrete time steps. In our case, the number of underlying hidden states ,$\mathbf{S} = \{s_t\}_{t=0..T_i}$, is unknown and the objective is to find the number of hidden states and then the sequence of them. Hence, we employed the sticky hierarchical Dirichlet process (HDP)-HMM in (Fox et al. (2008)). HDP-HMM uses Bayesian non-parametric methods in HMM to find the optimum number of states and their state transition matrix.

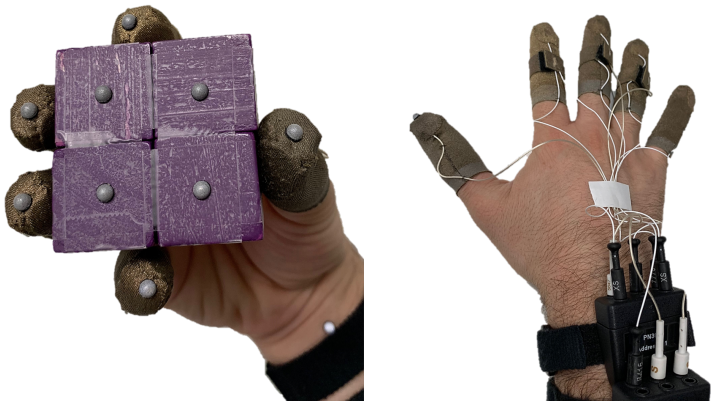


Figure 5.17: Setup for recording data to learn from human demonstration. Markers are attached to the object and fingertips to track positions in the motion-capture system (*OptiTrack*), and TPS sensors for recording exerted pressure on fingertips.

Model Training

We applied HDP-HMM on the recorded $\mathscr{D}_o$ of object's rotations (finger gaiting), where we stored fingers' relative angular velocities ω and pressures p as observations with ${}^o\boldsymbol{\psi}_t^{(i)} = [\omega_1^T, p_1, \cdots, \omega_4^T, p_4]^T$. Subsequently, an HMM model is derived over 100 computation trials of 2000 iteration step. The HMM model extracted 3 hidden states (see Figure 5.18), each corresponding to a task segment. From Figure 5.18, we observe that some fingers (index and thumb) contribute to task of object rotation, mostly in one segment, *"contributing"* state, and that, in the other segments, these fingers update their position, *"updating"* state.

Planning

According to task segmentation with HMM, this finger-gaiting manipulation consists of several fingers *attaching* to and *detaching* from the object's surface. In the phases of manipulation, where a finger does not interact with the object, it has to be controlled to preserve or increase its manipulability. Fingers that reach workspace limits will update their position in order to increase manipulability; and the other fingers will provide support to stabilize the object. In each task segment and for each finger, we detect the finger role, based on relative displacement:

$$\ell_i(s_t) = \begin{cases} 1, & \text{if } \bar{\dot{\boldsymbol{x}}}_i(s_t) < \epsilon \\ 0, & \text{if } \bar{\dot{\boldsymbol{x}}}_i(s_t) \geq \epsilon \end{cases}$$

where $\bar{\dot{\boldsymbol{x}}}_i(s_t) = \sum_{k=1}^{n_{s_t}} \|\dot{\boldsymbol{x}}_i^k\|^2 / n_{s_t}$ is relative mean velocity of i-th finger in task segment s_t, and $\epsilon > 0$ is a small positive value. Thus, $\ell_i(s_t) = 1$ indicates that i-th finger is contributing to rotation by inserting force on the object and $\ell_i(s_t) = 0$ denotes that for the given s_t finger i is detached. In case there is no interaction with the object, meaning that $\ell_i(s_t) = 0$, we compute

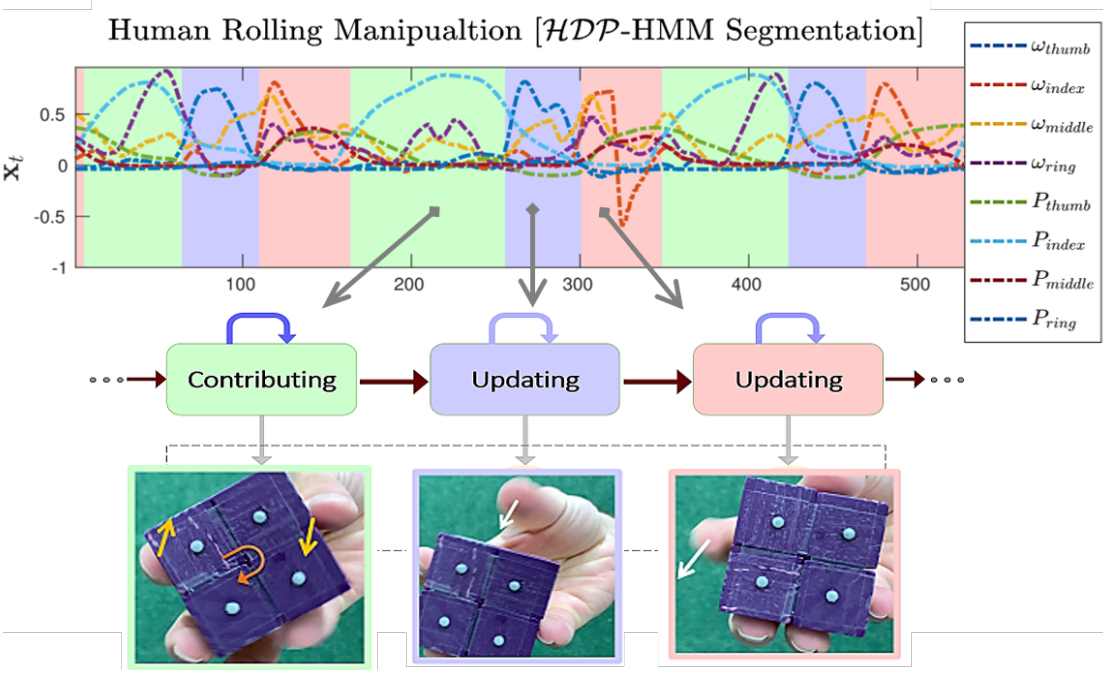


Figure 5.18: Task segments extracted from HDP-HMM using relative angular velocities and pressure data. The green segment corresponds to a state where some fingers, here index and thumb, “contribute" to object rotation. They remain in contact with the object, with increased contact forces and no relative angular velocity. The next segment is an “updating" state for a finger, here the thumb. The finger moves to get to the next position as its relative angular velocity increases, whereas the other fingers remain in contact to support the object. Similarly, the other segment is an “updating" state for other “contributor finger," index.

the finger's desired attractor $\boldsymbol{x}_i^{(2)}$ from Ψ_o :

$$\boldsymbol{x}_i^{(2)}|_{s_t=0} = \frac{1}{n_{s_t}} \sum_{k=1}^{n_{s_t}} \Big(u(\|\dot{\boldsymbol{x}}_i^k\|^2 + \delta) - u(\|\dot{\boldsymbol{x}}_i^k\|^2 - \delta) \Big) \boldsymbol{x}_i^k$$

here, $u(.)$ is the unit step function and $0 < \delta \ll 1$ is a small positive value. The extracted attractors for each task segment, and the desired object velocity are used for control and planning identical to *Section 5.5.2*.

Evaluation

We realize a *human like* in-hand object rolling (see Figure 5.15). Figure 5.16 shows an implementation of this approach on a real robotic hand. The object is grasped in a configuration suitable for both stability and for generating the desired motion (see *Section 5.5.1)*. The hand then begins rotating the object through a combination of several contact/non-contact movements; see also videos in the supplementary materials that explain a full cycle of rolling manipulation in different phases of finger motions. For this experiment, we could not use

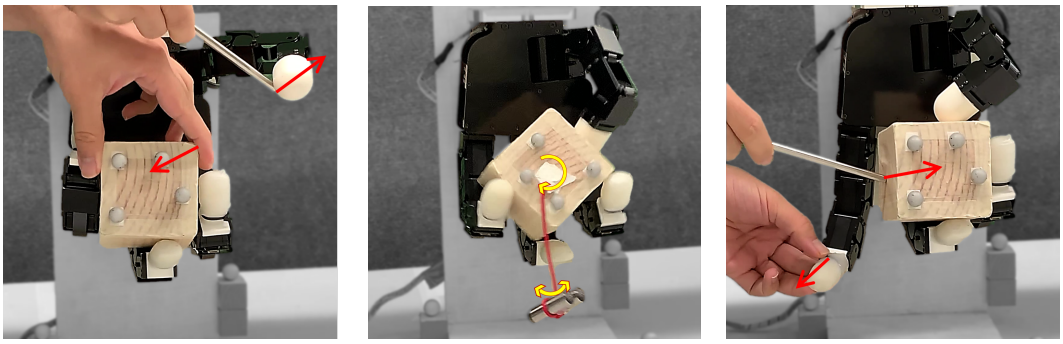


Figure 5.19: Testing the robustness of our control approach by applying various perturbations to the fingers and the object during the finger-gaiting task: (left) perturbing thumb, (middle) attached pendulum that weighs as 20% of the cuboid, generates dynamic perturbations during manipulation, and (right) perturbing object and pinky finger.

tactile sensors due to hardware limits and tactile sensor shape restrictions. Hence, to extract contact frames, we relied on prior knowledge of the object's shape and the robot's forward kinematics.

Table 5.3 summarizes the results of 10 replicates of this experiment for a full cycle of rolling object. Each cycle consists of 4 sequential repetition of the task shown in Figure 5.16. Our results in Table 5.3 show 85% success rate in task execution and 60% success rate in the full-cycle rotation (accomplishing 4 tasks in succession). Failures were in the last rolling executions and were mainly due object blocking a finger (not allowing for finger workspace update) and object slipping during rotation.

Table 5.3: Number of successful trials in 10 replicates of full object rolling experiment. Each cycle consists of 4 repetition of the rolling task in succession; see Figure 5.16 as an example of one rolling task.

Order	**I**	**II**	**III**	**IV**
Trials	10	10	8	6

Robustness

This complex manipulation shows that with appropriate high-level planning, our control framework enables the encoding of a robust solution for task execution, even in face of different types of disturbances; see Figure 5.19. Disturbances were tested in three scenarios: (a) we add an additional unknown mass dangling from a wire, hence modifying significantly the mass and mass distribution, (b) we move the object horizontally, to showcase the adaptation of the fingers, and coordination and the immediate re-positioning of the fingers relative to the object's location, and (c) we pull one finger away and subject the object to an external force to validate the adaptation of the forces at the fingertips that re-balance the object in the absence

of one finger. See also a supplementary video of the experiment that shows examples of rolling in-hand manipulation and efficient adaptation of our controller to various perturbations.

5.7 Discussion and Summary

In-hand manipulation and dexterous grasp are among the challenging tasks for real-world robot applications. It is not straightforward, if not impossible, to acquire a universal framework for grasping different objects and for performing various manipulations. Due to the diverse complexities, most real-world implementations of in-hand manipulation are engineered for a specific task. The choices of tasks, objects, and the manipulation scheme are constrained due to hardware limits. Nevertheless, we achieved several dexterous grasp and manipulation focusing on post-grasp stability and manipulation accuracy. In this chapter, we centered our attention on the robustness and adaptation, accordingly. Hence, all of our evaluations are on a real-world robotic setup, where the uncertainties and modeling challenges are significantly higher than the simulated environment.

In this chapter, we have introduced a new control framework for grasping and manipulating objects. The approach of this study is based on synchronizing finger (task planning) and adapting joint torque (low-level control). We first begin with robotic-hand control. During any dexterous manipulation, we will not be able to hold an object securely if the robot hand system itself is not entirely stable. Therefore, learning and compensating for the dynamics of the robotic hand is of primary importance. However, precise knowledge of the physical properties of the object and the robot is never available. Therefore, we developed a compliant torque controller rather than using stiff joint positions. Our proposed coordinated adaptive torque-controller was designed to learn the robot dynamics for precision and compliance. We have compared our torque controller against a widely utilized low-level controller (PD) and have shown that our approach is consistently more accurate while maintaining finger joints compliant and coupled.

Regarding grasp and manipulation, our main intuition is to acquire a coordinated multi-finger system and to develop a method for increasing robustness under uncertainties, such as varying or uneven mass distribution. We assume that, for each object, the grasp configuration is given and is not necessarily optimal or well-calibrated. We consequently improve this prior knowledge through tactile sensing and an online re-estimation of the contact frames. This enhanced knowledge, in addition to the robot's kinematic feedback, facilitates modulating object-level impedance for a more secure grasp. The effective frequency of the tactile sensor, however, restricts the update rate of the object-level impedance. As a result, the controller can tackle only the disturbances with a lower rate than this frequency.

More precisely, the proposed controller for grasp and manipulation uses contact-frame estimation (grasp matrix computation) and grasp optimization to construct an object-level impedance. We then use the devised coordinated adaptive torque controller to realize the desired impedance and perform grasp/manipulation tasks. Our main focus in this chapter

was the robustness of grasp and manipulation. Hence, the pipeline has been extensively tested in various real-world robotic experiments and performed with high accuracy and robustness, under different uncertain conditions. Finally, we have shown that our framework, along with learning from demonstration, can successfully perform complicated manipulations, such as finger gaiting.

To further improve the application of our approach, there are several limitations that remains a research problem to address. In most of the experiments, slippage, especially for heavier objects, caused some trials to fail. Slippage detection and appropriate adaptation, accordingly, could improve the success rate. Although the object pose estimation within the hand by the OptiTrack system was accurate, it was never easy to obtain. Marker occlusion during finger motions occurs frequently, thus rendering pose tracking rough/unsteady. Seeking more reliable feedback, we would like to estimate the object's pose by using tactile observations within the hand frame.

6 Compliant Robotic Hand Controller in Human-Centric Environment

This chapter comprises two applications of our compliant robotic hand controller in human-centric environment. We work on dexterous capabilities for a robotic hand to expand its prehension and apprehension applicabilities in a human-centric environment.

At the time of writing, the work presented in Section 6.1 is submitted as *Khadivar, F., Mendez, V., Correia, C., Batzianoulis, I., Billard, A., and Micera, S. "EMG-Driven Shared Human-Robot Compliant Control for In-Hand Object Manipulation in Hand Prostheses," Journal of Neural Engineering, 2022.* Vincent Mendez is the co-first author of this paper. Vincent is the main contributor to Electromyography (EMG) decoding of forearm muscles to enable subjects to move, proportionally and simultaneously, the fingers of a robotic hand. He also contributed to inferring the desired object rotation using EMG electrodes that record shoulder elevation and depression. The methodology of these contributions is presented in Appendix C.1. Section 6.1 presents the EMG-Robot interface, the robotic compliant controller (for both the robotic hand and the robotic arm), and the statistical analysis developed by the author of the thesis.

The work presented in Section 6.2 is prepared to be submitted as *Khadivar, F., Yao, K., Gao, X., and Billard, A. "Online Active and Dynamic Object Shape Exploration with Multi-fingered Robotic Hand," IEEE Robotics and Automation Letters (RA-L), 2023.* Kunpeng Yao is the co-first author of this paper. Kunpeng is the main contributor to optimization-based planning algorithm that online adapts the hand pose to the local surface geometry. The methodology and results of his contributions are presented in Appendix C.2. Section 6.2 details the theoretical advancement and simulated evaluation of the active exploration strategy accomplished by the author of the thesis.

6.1 Shared-Control for In-Hand Manipulation with a Myoelectric Prosthesis

Commercial prostheses can perform a reasonable number of grasps; however, they are often inadequate for manipulating the object once in hand. The lack of dexterity drastically restricts the utility of robotic prosthetic hands(RPHs). Shared-control approaches that give some authority to an autonomous controller on-board the prosthesis is an alternative that may improve the dexterity and versatility of RPHs for people with trans-radial amputation. Here, we aim to investigate a novel shared control strategy for robust object manipulation that combines autonomous control of forces exerted by the robotic hand with EMG motion decoding.

With a look towards the availability of better and more performing hand prostheses – and while recognizing that individuated finger motion decoding from EMG will remain limited in its accuracy – we design shared control mechanisms where a robotic controller is in charge of low-level closed-loop control. This controller enables online adaptation of the positioning of fingers and stabilizing the object in hand through forces applied by the fingers. To this end, we build on advances made in this thesis for controlling a robotic hand to enable human-like control of the fingers in either individual or coordinated manner. Such enhancements permit the execution of robust grasps with multi-finger robotic hands.

We test novel shared control strategies to perform EMG-driven insertion tasks. We show that this shared control approach allows subjects to perform a continuum of manipulation: from grasping an object to manipulating it in air and during insertion when in contact with another object. The subject remains in control of deciding when to grasp and release the object and how to preshape the hand. Importantly, our shared control mechanism enables subjects to perform in-hand object rotation when controlling a dexterous 16-DoFs robotic hand. This is achieved thanks to the following:

(i) Single-finger proportional decoding to infer high-level motor intentions; in-hand object rotation is decoded via elevation or depression of the shoulder recorded from EMG signals. Vincent Mendez is the main contributor to this part which is presented in Appendix C.1.

(ii) A virtual object-based compliant controller for low-level robot control. Given the task objective and tactile feedback, the low-level control of the robotic hand employs an autonomous and adaptive compliant controller that regulates online the interaction forces at contact points.

(iii) An interface that uses feedback-based state machines to integrate high-level and low-level robot control. The interface selects which task has to be executed, provided the subject's command and robotic feedback.

We evaluate our shared control approach in an experimental setup by conducting a 3-days long longitudinal study with healthy individuals. Motor intentions are decoded from the

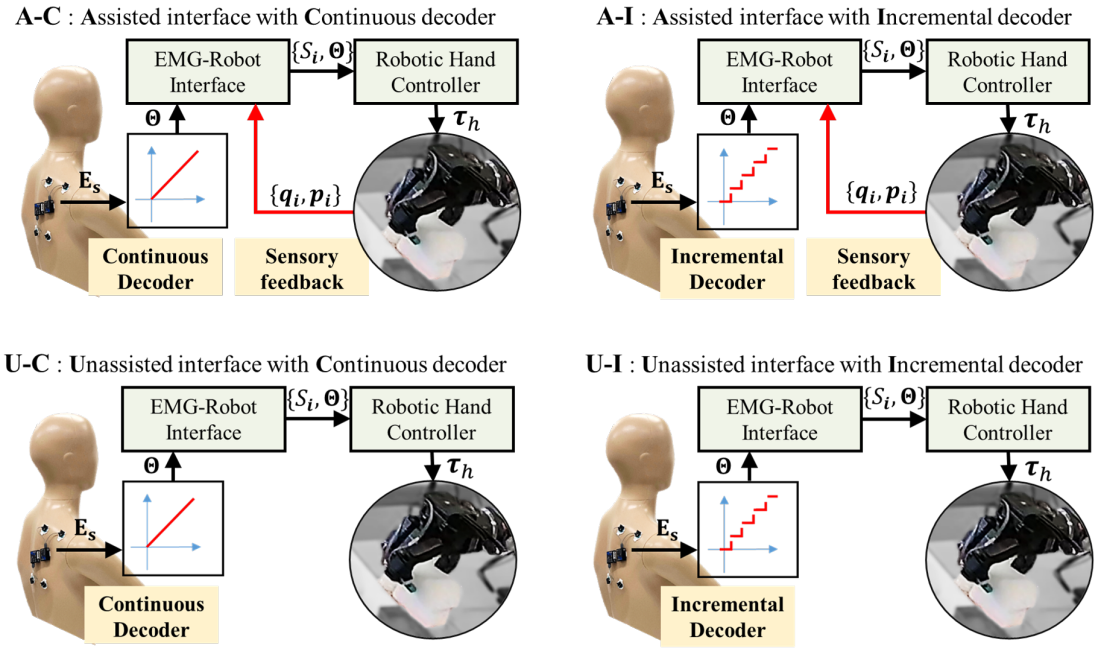


Figure 6.1: List of shared control conditions. We conduct a 2x2 experimental study with two levels of autonomy in the EMG-Robot interface, (**U**) unassisted and (**A**) assisted, and with two types of EMG shoulder decoding, (**C**) continuous and (**I**) incremental. $\mathbf{E_s} \in \mathbb{R}$ is the EMG signal from the shoulder. Θ is a vector of high-level commands determined via EMG decoding (see Figure.6.2). $S_i \in \mathbb{R}$ denotes the state machine defining the hand's control mode (see Figure.6.5), and $\boldsymbol{\tau}_h \in \mathbb{R}^{16}$ is the joint torques computed to control the robotic hand. $\boldsymbol{q}_i \in \mathbb{R}^4$ and $\boldsymbol{p}_i \in \mathbb{R}$ are, respectively, the joint positions and tactile sensor feedback for the i-th finger.

forearm and the shoulder's EMG signals. We hypothesize that a shared control scheme with more robot autonomy would result in more robust object manipulation. In contrast, we expect that more user control would allow the subjects to rotate the object faster.

6.1.1 Task Introduction

We consider the task of inserting an object into another object. Such a "peg in hole" task can be considered a robotic benchmark. While it has received attention already in the 70's (Takeyasu et al. (1976)), it remains a topic of research (Tang et al. (2015)). As simple as it may appear, inserting an object into another one still relies on complex algorithms to determine when the object is jammed and when to correctly adapt the object's orientation so that neither object nor the robot is damaged. Such rapid and compliant control of finger-object interaction requires estimating the force at contact and adapting fingers' motion accordingly. Contact detection is usually done through reading of tactile sensors, sometimes in combination with vision (Karayiannidis et al. (2016)).

When controlling a prosthesis tasked to insert an object into another one, we must assume

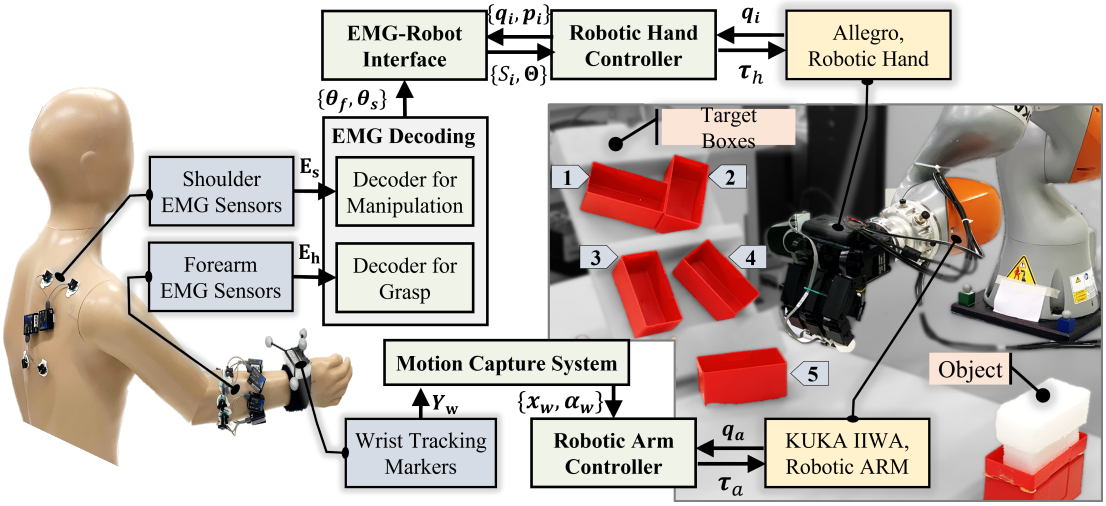


Figure 6.2: Overview of the experimental setup. The subject is asked to grasp a cuboid and place it in one of the target boxes, differing in position and orientation. The order of all pick and place tasks, as specified from 1 to 5, is fixed throughout the entire recording. A KUKA iiwa 14 robotic arm is set to follow the displacement of the subject's wrist acquired via the OptiTrack capture system ($\mathbf{Y}_w$: markers tracking signals, $\boldsymbol{x}_w \in \mathbb{R}^3$: wrist position, $\boldsymbol{\alpha}_w \in \mathbb{R}^3$: wrist orientation, $\boldsymbol{q}_a \in \mathbb{R}^7$: joint position feedback of the robotic arm, and $\boldsymbol{\tau}_a \in \mathbb{R}^7$: joints torque command send to the robotic arm). A left Allegro robotic hand mounted with BioTac tactile sensors executes grasp and manipulation received from the EMG decoder. The hand is commanded to open or close via processing forearm EMG signals, and to rotate the object within the hand frame through shoulder EMG activation ($\mathbf{E}_h \in \mathbb{R}^{200\times6}$: forearm EMG signals, $\mathbf{E}_h \in \mathbb{R}^{200\times2}$: shoulder EMG signals, $\boldsymbol{\theta}_f \in \mathbb{R}^5$: scaled fingers flexion, and $\boldsymbol{\theta}_s \in \mathbb{R}$: scaled shoulder flexion).

that the autonomous controller of the prosthesis has access solely to tactile information. Corrections driven by visual feedback are conveyed implicitly through EMG-based intention detection, the latter driven by the amputee's visual appraisal of the situation. The designed framework here uses an autonomous compliant control of fingers to adapt fingers' orientation and exerted forces based on tactile information only.

6.1.2 Approach

We compare four different shared control schemes; see Figure 6.1. We perform a 2x2 experimental study with two levels of autonomy in the state-machine interface versus two types of EMG shoulder decoding, resulting in a total of four types of shared control schemes. One shoulder-decoding approach is incremental, and the other one is continuous.

The incremental one is threshold-based decoding. When the subject is elevating or lowering the shoulder, the decoder modulates the rotation value and remains constant when the shoulder is resting. The continuous decoder is based on a support vector regression (SVR) algorithm that directly maps the shoulder position to the rotation value. We envision that

the continuous decoder would allow the subject to rotate the object faster (Cler et al. (2014)), whereas the incremental one is more robust to noise.

The two autonomy modes of the state-machine interface are (i) unassisted and (ii) assisted. In the former, the controller relies only on the command and state sequence, whereas the latter also uses feedback from the robotic system. The assisted mode utilizes tactile sensing and finger kinematics for identifying the task and its status. We expect the assistance to reduce trial failures because it would avoid the unexpected drop of objects coming from noise in the EMG decoder.

In an experimental setup, we evaluate the functional gain of our approach and compare the performance of the proposed shared controllers in a dexterity test. We adopt the “Grooved Peg Test" (Wilcox and Nordstokke (2022)), a standard dexterity assessment. The test requires fine motor skills to place grooved pegs in holes with different orientations. This test is performed routinely to quantify the development of dexterity in 6-year-old children (Wilcox and Nordstokke (2022)), and the loss of dexterity in stroke patients (Thompson-Butel et al. (2014)). We adopt the Grooved Peg test to the size of the robotic hand at our disposal and design a peg-in-hole task where subjects had to grasp a rectangular object and place it in boxes with different angles (Figure 6.2). An inclined surface was used to avoid the subjects relying on gravity to put the object in the box when the angle of the object did not perfectly fit the angle of the box.

Experimental Protocol

The experimental protocol is summarized in Figure 6.4. Subjects participate in 3 sessions consisting of 4 experiment runs. In each run, a specific control condition is selected. Given the control condition, subjects perform 5 tasks of picking and placing a cuboid in different target boxes. In our experimental protocol:

(i) A task consists of grasping a cuboid and inserting it in a target box, see Figure 6.3. Each experiment run consists of 5 tasks that differ only in position and orientation of target boxes; see Figure 6.2. Throughout the entire experiment, the order of tasks is fixed as given in Figure 6.2.

(ii) There are 4 control conditions specified as the combination of 2 shoulder control conditions (incremental and continuous EMG decoder) and 2 state-machine interfaces (assisted and unassisted). Control conditions are listed in Figure 6.1, and explained in detail in Appendix C.1.1 and Section 6.1.2. To eliminate the effect of factors like fatigue in our analysis, the order of control conditions is chosen randomly for each subject and session.

(iii) Before each session, subjects have to calibrate the finger decoding model and the continuous shoulder controller. The calibration process takes between 12 to 15 minutes; see Appendix C.1.

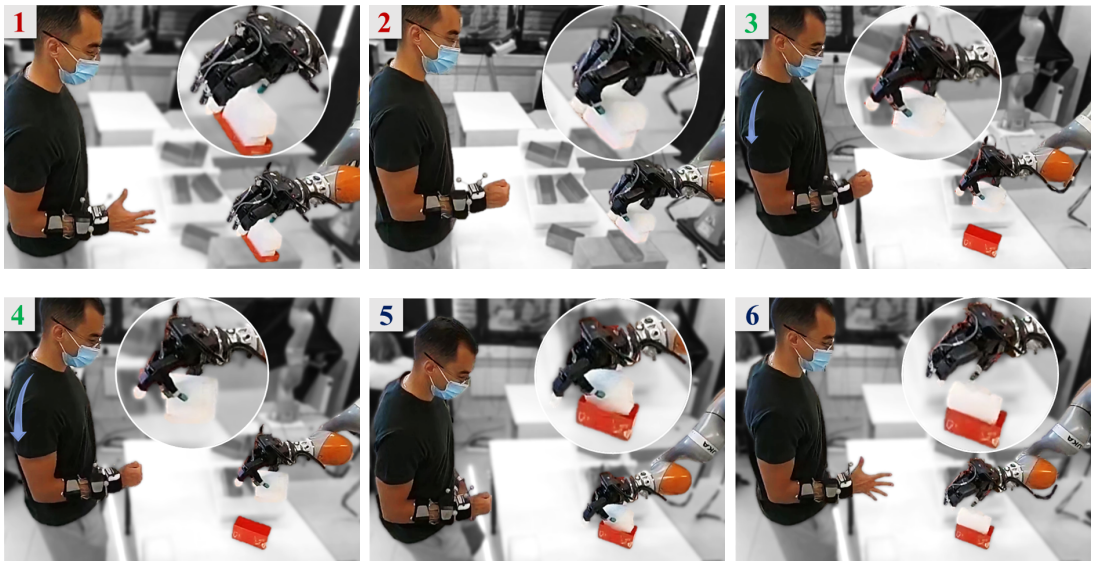


Figure 6.3: An example of one experiment run. The subject first grasps the object (red numbers), then rotates it within the robotic hand (green numbers), and finally inserts it in the target box (blue numbers). (1) the subject needs to bring the robotic hand to the picking position. The robotic hand is in *open hand* mode given the fingers' EMG decoding. (2) the user commands the robotic hand to grasp the object by closing his fingers. Then, he lifts the object and moves the robotic arm near the target box. The robotic hand is autonomously holding the object. (3) Through shoulder movement, the subject tries to align the orientation of the object and the target box. (4) the subject aligns the orientations by rotating the object within the robotic hand frame. (5) the object is placed in the target box and then released (6).

(iv) After model calibration, subjects are verbally instructed to perform the experiment. Subjects are not informed which control condition is utilized. After each experiment run with a control condition, subjects are given a 2-minute break before starting with the next control condition.

(v) Since familiarizing with the setup can affect the performance progress, all subjects participated in the experiment in 3 sessions over 3 consecutive days in order to account for the expected learning effects.

(vi) An experimental session takes between 60 to 100 minutes for all subjects.

Subjects and EMG Recording

Eight subjects are recruited for the second study, all of whom are males aged between 26 and 30 (average weight: 74kg, average height: 177cm, and two left-handed). A Noraxon DTS system wirelessly records EMG signals at 1 kHz. Six bipolar surface EMG channels are placed on the right forearm to target the extensor digitorum, flexor carpi radialis, palmaris longus, flexor digitorum superficialis, and flexor carpi ulnaris muscles, located with palpation. Two other

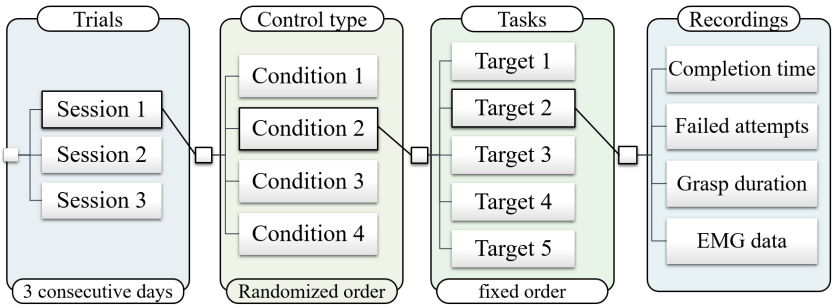


Figure 6.4: Summary of the experimental protocol. In 3 consecutive days, a subject participates in 3 sessions of 4 experiment runs. In each run, a specific control condition among 4 others is selected. The order of control conditions is randomized in every session and for all subjects. After setting the control condition, subjects complete 5 pick-and-place tasks, differing in the target positions. The task order is fixed from the top left box to the bottom right one in Figure 6.2. During task execution, we record completion time, the number of failed attempts, the grasping time duration, and EMG signals.

bipolar surface EMG channels are placed on the right shoulder and the back of the subjects to target the upper and lower fibers of the trapezius that allow, respectively, elevation and depression of the shoulder joint.

EMG-Robot Interface

The robotic hand operates in three modes:

(i) *Preshape*: robot opens and closes each finger individually.

(ii) *Grasp*: robot grasps the object by closing the fingers and secures contacts using tactile sensors.

(iii) *Manipulate*: robotic fingers manipulate the object while holding it in the hand frame. Fingers move in a coordinated object-centered fashion to rotate the object.

We model each of these modes via a set of state machines, $S = \{S_1, S_2, S_3\}$, see Figure 6.5 where S_1, S_2,and S_3 represent *preshape*, *grasp*, and *manipulate*, respectively. Transitions from S_2 to S_3, and from S_3 to S_1 are unidirectional. However, the switch from S_1 to S_2 is bi-directional, meaning that if the grasp is not sound, the subject can re-open the hand and attempt a new grasp. The bi-directional transition between these two states allows the subject to open/close fingers several times to find the seemingly fine grasp.

We define and evaluate two state-transition functions for our interface. One takes only the EMG command as input (unassisted interface), and the second one uses robotic feedback in addition to the EMG command (assisted interface), see Figure 6.5. More precisely, in the assisted interface, we benefit from tactile feedback to verify contact with the object and grasp

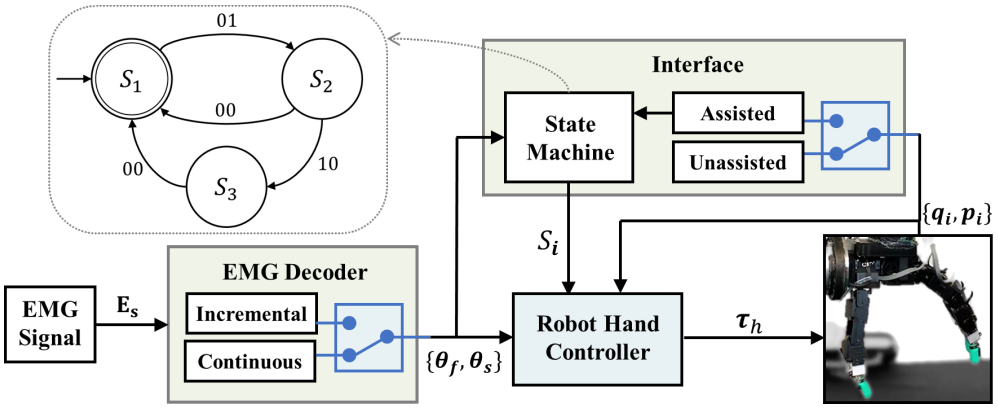


Figure 6.5: Block diagram of robot hand control with state machines of the (shoulder) EMG-robot interface. S_1, S_2,and S_3 represent *preshape*, *grasp*, and *manipulate* modes respectively. We assess 4 control conditions for controlling the robotic hand. Control conditions are the combination of 2 EMG decoders (incremental/ continuous) and 2 interface modes (assisted/ unassisted)see Figures 6.1 and 6.2.

realization, used in the transition from S_2 state to S_3 state. Moreover, from finger kinematics and the joint angles, we estimate object angular displacement. This estimation is then used to confirm the attainment of the desired motion. Once the desired manipulation is achieved, the robotic hand releases the object only if the command from the subject is received consistently over a time window of 4 s, e.i. the subject insists on opening the hand.

6.1.3 Autonomous Robot Controller

We control the robotic arm and hand in the torque mode, maintaining joint compliance. Compliant controllers enable the robot to adapt to perception uncertainties and provide safer human-robot interaction compared to position controllers. We introduced an additional safety layer on top of the controller, a user-robot collision avoidance that enhances the safety of human-robot interaction. Our autonomous compliant controller has two sub-modules: (a) a controller to imitate the subject's wrist displacement in position and orientation with the robotic arm end-effector (EE), and (b) a controller developed in the previous chapter that computes the joint torques of the robotic hand to perform tasks like grasp and manipulation.

Robotic arm control

The desired position and orientation of the robotic arm EE are computed based on the measured subject's wrist motion. From the OptiTrack system, we obtain the $3D$ position (x, y, and z axes) and orientation (roll, pitch, and yaw) of the subject's wrist with respect to the world frame. In our tracking strategy, the EE must mimic the subject's wrist displacement in 4 coordinates, $3D$ position, and the angle in the roll axis. Since the subject and the robot are facing each other in our setup, the robot EE is commanded to mirror the displacement in the

pitch axis of the subject's wrist. To restrict the object rotation only to in-hand manipulation in our control scenario, the robot EE is indifferent to the displacement in the yaw axis of the subject's wrist achieved with Rodrigues' rotation formula (Belongie et al. (1999)). We found that mirroring the pitch orientation and not tracking the yaw rotation is more intuitive for subjects and results in more accurate tracking simultaneously.

The desired pose of the robot EE is passed to a predefined linear dynamical system to find the desired translational and angular velocities. These velocities serve as the inputs for our underlying compliant and passive controller (Kronander and Billard (2015); Khadivar et al. (2021a)) that outputs a set of joint torques for the robotic arm. Thanks to the dynamical system approach, our controller is robust to perturbations and disturbances and safe for human-robot interactions due to compliant passivity. The redundant DoFs of the robotic arm are constantly optimized to decrease robot acceleration while traversing from one point to another.

Robotic hand control

Similar to the robotic arm control, this controller also requires a set of desired fingertip positions/ velocities as inputs. The computation of these inputs depends on the task state and whether or not a finger is in contact with the object. In the pre-grasp state, the desired position is computed based on the angle from the EMG decoder. In this state, each finger of the Allegro hand is separately commanded by the EMG decoder. The object's relative position with respect to the robotic hand is computed by tracking the subject's wrist motion. Once the object is within the hand frame, subjects changing hand posture from open hand to fist is equivalent to a robotic hand attempting to grasp. From tactile feedback, the control state changes from pre-grasp to grasp if three fingers make contact with the object. In this state, the fingers establish a force-closure (Prattichizzo and Trinkle (2016)) around the object. In other words, in case the subject feels a higher grip force is needed to obtain a firm grasp and lift the object, it can be achieved by clenching the fist tighter.

In the manipulation state, where the fingertips are in contact with the object, the control inputs are determined based on the object's desired position. In the grasp state, the object's desired position is fixed. When the subject intends to rotate the object within the robotic hand, in the manipulation state, the object's desired position changes, given the input from the EMG shoulder decoder. This desired pose is relative to the palm frame (a frame attached to the base of the robotic hand) and is used to find the desired object velocity through a linear dynamical system. Translating an object-centric desired velocity to individual in-contact fingers' desired velocities is realized through grasp matrix transformation (Prattichizzo and Trinkle (2016)), grasp stability metric, and the estimated contact mechanical properties. More precisely, we obtain the desired velocity of all fingers from the desired object velocity commanded by the shoulder EMG decoder. These velocities, similar to the robotic arm control, are sent to our underlying compliant and passive controller to compute the corresponding joint torques.

6.1.4 Results

Shared-Control Functional Assessment

For each of the 5 different tasks within an experiment run, we measured the completion time and the number of failed attempts. To better follow the results, the key terms and main outcome variables (completion time and number of failures) are listed below, following control conditions in Figure 6.1.

(i) *Control condition*: There are 4 control conditions differing in the type of employed state-machine interface and EMG decoder.

(ii) *Session (Trial)*: A series of 4 experiment runs where the subject completes all tasks with all control conditions. Each subject participates in three sessions over three days.

(iii) *Target*: A box (hole) in a specific pose where the object (cuboid) is placed. In each experiment, there are five fixed targets placed in different positions and orientations.

(iv) *Task Completion Time (success time)*: The time duration in a successful trial that it takes to lift the object from grasp position and place it in the target box. In other words, the time to finish a manipulation/insertion of a peg in a hole. This time duration starts from the moment that the subject lifts the object.

(v) *Number of Failed Attempts*: The number of failed manipulation attempts. An attempt is counted as a failure if the object is not fully placed in the specific target.

On average, one session for a subject took 75 minutes. Approximately, 30 minutes were used at the beginning to place the EMG electrodes and calibrate the finger, as well as the shoulder decoders. Then, performing the 5 pick and place tasks took on average 6 minutes and 11 *s* for each condition. Figure 6.6 provides an overview of the collected data, for all participants and conditions.

Statistical analysis was performed to test for significant changes in performance (in terms of the number of failures and completion time) across the different sessions, control conditions, and targets. To check if the variables are normally distributed, we use the Shapiro-Wilk test, verifying that both the completion time and the number of failures are not normally distributed. Hence, we test for significant changes in performance by using non-parametric statistical tests.

Task Performance across Sessions

Figure 6.7 shows the mean number of failed attempts and mean completion time for the 8 subjects, across the 3 sessions. Using the Friedman's ANOVA test, we observe a statistically significant difference in task completion time across the 3 sessions, $\chi^2(2) = 31.962, p < 0.001$.

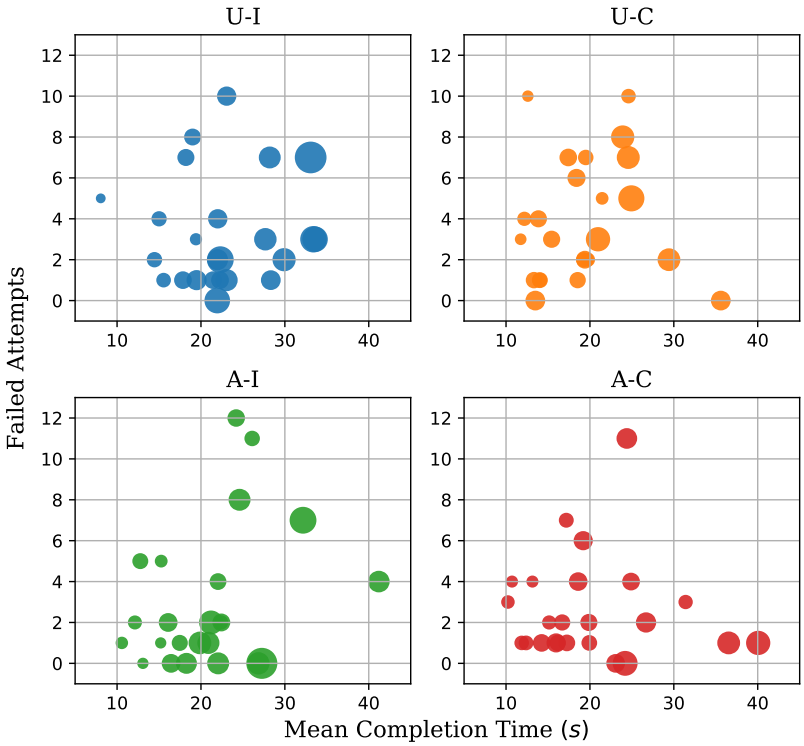


Figure 6.6: All collected data from all 8 participants. Each subject tried 4 different shared control conditions (see Figure 6.1) in 3 sessions over three days. Thus, for each control condition, there are 32 data points (8 subjects ×3 sessions). In each experiment run, subjects were asked to complete five sub-task in a specific order (Figure 6.2), placing a cuboid in five different target boxes. We recorded the number of failed attempts and the completion of each sub-task. The x-axis for each plot is the mean completion time of these five sub-tasks, the y-axis is the corresponding sum of the failed attempts, and the size of the marker indicates the relative variance in completion time over.

Post hoc analysis with Wilcoxon signed-rank tests was conducted by using a Bonferroni correction, setting the significance level at $p < 0.05/3 = 0.017$. We observe no significant differences between Session 1 and Session 2 ($Z = -1.943, p = 0.052$), whereas there are significant changes in completion time between Session 1 and Session 3 ($Z = -4.996, p < 0.001$) and Session 2 and Session 3 ($Z = -4.281, p < 0.001$). In fact, from Session 2 (median= $18.0s, IQR = 11.3$) to Session 3 (median = $15.2s, IQR = 10.2$) there is a significant decrease of $2.8s$ in median completion time, and from Session 1 ($median = 19.6s, IQR = 13.4$) to Session 3 the decrease is of $4.4s$.

For the number of failures, we also verify a statistically significant difference between the 3 sessions using the Friedman's ANOVA, $\chi^2(2) = 11.494, p = 0.003$. Using the Wilcoxon signed-rank test we see, (also in Figure 6.7), that only from Session 1 (mean = 0.96, standard deviation (SD) = 1.5) to Session 3 (mean = 0.41, SD = 0.68) the mean number of failures decreases significantly ($Z = -3.639, p < 0.001$) by an average of 12.1 failures. From Session 1 to Session 2

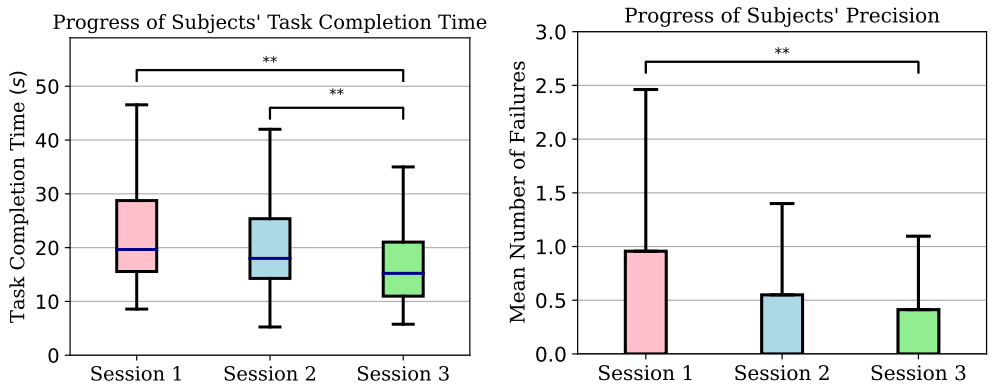


Figure 6.7: Performance progress of all subjects and all control conditions over 3 sessions (3 days). Left: boxplot of task completion time per session; midlines indicate the median. Right: barplot of mean number of failed attempts for each session. Statistically significant differences regarding mean completion time can be seen between sessions 1 and 2, and sessions 2 and 3. For the mean number of failures, only the first and third sessions differ significantly. Error bars: $+SD$, $**: p < 0.01$.

($Z = -2.321, p = 0.020$) and from Session 2 to Session 3 ($Z = -1.509, p = 0.131$) there are no statistically significant changes.

Comparison between Shared Control Conditions

We investigated the performance of four shared control schemes, which combined two state-transition modes (assisted vs. unassisted) with two EMG decoding approaches (continuous vs. incremental). The mean task completion time and mean number of failures were obtained and analyzed for each of the control conditions, for all subjects and experimental sessions (see Figure 6.8).

Using again the Friedman's ANOVA test with $\alpha = 0.05$, we observe a statistically significant difference in task completion time between the 4 experimental conditions, $\chi^2(3) = 11.325, p = 0.010$. The same is verified for the number of failures, $\chi 2(3) = 10.305, p = 0.016$. By running the Wilcoxon signed-rank test with $p = 0.05/4 = 0.0125$, we find that the task completion time in condition U-I (median = 19.0s, IQR = 11.9) was significantly longer than the completion time in U-C (median = 17.4s, IQR = 10.4), $Z = -2.803, p = 0.005$. That is, without assistance, subjects were significantly faster by an average of 3.36s using the continuous controller, when compared to incremental control. Between U-I and A-I (median = 16.9s, IQR = 12.2) – i.e, the two assistance modes using the incremental EMG decoding –, we observe that the assisted robotic interface led to a significantly faster performance ($Z = -2.71, p = 0.005$), by an average of 2.48s.

Regarding the number of failures, we see statistically significant changes between U-I (mean

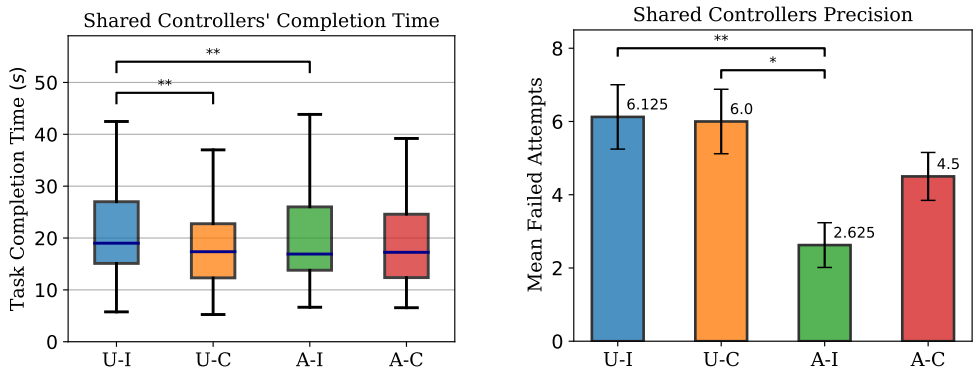


Figure 6.8: Performance of all shared controller conditions across all subjects and sessions. Left: boxplot of task completion time across control conditions. Right: barplot of subject's mean number of failed attempts per control condition. Error bars: $\pm SD$, $* : p < 0.05$, $** : p < 0.01$.

6.125) and A-I (mean 2.625), $Z = -3.046$, $p = 0.002$, and between conditions U-C (mean 6.0) and A-I ($Z = -2.549$, $p = 0.011$). In fact, the number of failed attempts for both control schemes that used an unassisted interface significantly improved with the assisted interface, when combined with the incremental control modality (by 3.5 and 1.5 difference, respectively).

Overall, comparing the two state transition modes (unassisted vs. assisted), the assisted interface was significantly better than the unassisted counterpart, in terms of resulting in lower number of failed attempts (41% more precise) (Wilcoxon test, $Z = 3431.5$, $p < 0.05$). Regarding completion time, no significant difference were observed between the two methods. On the other hand, the two EMG decoding approaches (incremental vs. continuous) performed similarly in terms of both completion time, regardless of the assistance mode.

Taking a closer look at the performance for each of the 5 targets, Figure 6.9 shows that the assisted mode resulted in significantly lower failures, especially for those that require a large change in wrist orientation and large in-hand rotation (tasks 1,2, and 4 where the necessary angle is higher). The first two tasks are considered the most difficult ones since the subject needs to adjust the orientation of the robot's EE (palm being parallel to the target box inclination) and rotate the object in hand simultaneously.

Results Summary

In summary, the assisted control mode significantly improved performance by lowering the number of failed attempts. The assisted state transition mode, when in combination with EMG incremental control condition, *A-I*, outperformed the others as it has the lowest completion time and significantly lower number of failures (higher precision). On the contrary, the incremental EMG decoding without assistance (U-I) proved to be the least efficient among control conditions. Comparing the control condition *U-C* with the lowest robot autonomy

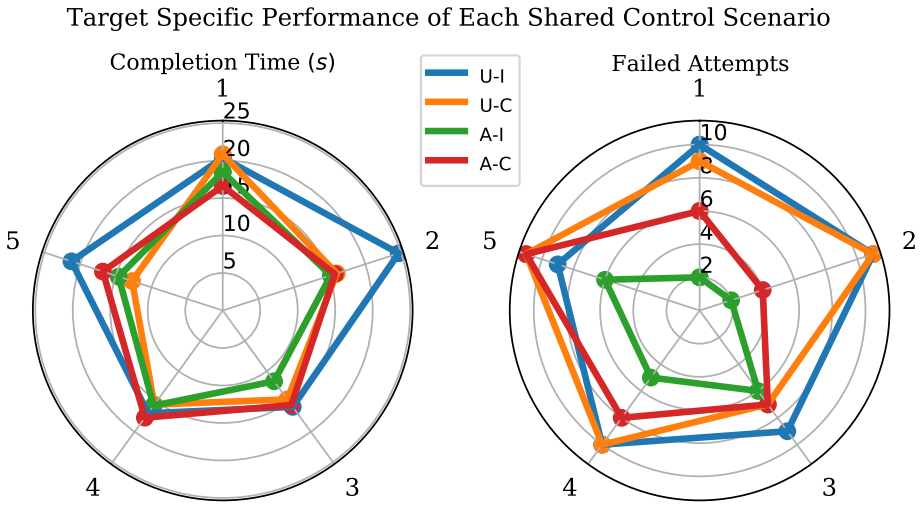


Figure 6.9: Median completion time (left) and Mean Failed attempts (right) for each pair of sub-tasks and shared control condition. In each experiment run with a set shared control condition, subjects had to place a cuboid in five different boxes. Here is shown the overall performance of each control condition in accomplishing a specific sub-task.

with the condition *A-I* with the highest robot autonomy shows that increasing the autonomy of the robot controller improved the precision (reduced number of failures) more significantly than the efficiency (task completion time).

6.1.5 Discussion and Summary

In this application, we used a robotic hand to pick, manipulate and place an object in various target boxes. We examined four shared control conditions based on a compliant controller in conjunction with EMG decoding to teleoperate a robotic arm while maintaining full autonomy over high-level commands. Given the experimental results, we propose the shared control strategy with assisted state-machine interface and incremental EMG decoder for the shoulder. We observed that the number of failed attempts and the completion time decreased over test sessions across all tasks and subjects (Figure 6.7). This confirmed that subjects learned how to improve their performance on this modified grooved peg test and did not rely on an increase in EMG decoding accuracy.

The target boxes were on an inclined surface, requiring the subjects to rotate their wrist. Although forearm orientation is a major source of noise for real-time applications (KyungYou et al. (2010)), the control technique developed in this study was robust enough for the subjects to complete the task. Calibrating the model in the various arm and wrist orientations could improve the robustness of wrist rotation and arm orientation (Park et al. (2016)), but this would imply a substantially longer calibration time.

To accomplish in-hand object rotation with a robotic hand, precise finger motions associated

with sensory input are necessary (Li et al. (2016a)). In this study, information such as object position, displacement, and grasp position is obtained directly from the subjects. Computing such variables and states is challenging when approaching complex manipulation tasks. The controller had to remain reactive to the user's commands with a minimal execution time delay to increase intuitiveness and sense of agency (Rognini et al. (2013)). At the same time, the controller had to be robust to inconsistencies coming from the EMG decoder due to the variability of the signals. We estimated the object's pose relative to the robotic hand from the forward kinematic and the fingers' joint position. This estimation proved helpful in these tasks with a cuboid object; on the contrary, for more complex object shapes and to obtain higher accuracy, more sophisticated object pose estimation methods like vision-based (Doosti et al. (2020)) or tactile images (Sodhi et al. (2021)) are required.

From Figure 6.9, target tasks that required larger in-hand rotation foreseeably took more time to be completed. Reducing delay in user command execution and increasing the decoding precision (Xia et al. (2018)) are instrumental for future enhancement to reach human-level performance. Furthermore, we can use the information from the robotic hand to provide sensory feedback to the users. For instance, contact, force, and finger angle information gathered from the robotic hand can be given to users with trans-radial amputation through invasive (D'Anna et al. (2019)) or non-invasive channels (Stephens-Fripp et al. (2018)). Sensory feedback can help users to send more accurate high-level commands to the robotic hands, improve embodiment (Bensmaia et al. (2020)), and reduce cognitive load (Valle et al. (2020)). Assistance from the robotic state became greatly beneficial for targets 1 and 2, where decoding the subject's intention and encoding the desired command was more convoluted. Indeed, the two inclined targets had a high angle and elevation. For these targets, the assisted state transition mode significantly decreased the number of failures compared to the unassisted interface (Figure 6.9).

Finally, We showed that combining a shared control condition with EMG decoding for finger motions and object rotation could be a realistic alternative for users with trans-radial amputation to improve the dexterity and versatility of RPHs. The shared control created in this study could allow users with an amputation to manipulate objects in their prosthetic hand, which is practical in numerous daily tasks. In our teleoperated system, subjects maintained complete control over the robotic hand. The condition that obtained the best results was the incremental shoulder EMG decoder with the assisted state transition mode. Surely, integration of an RPH and validation on people with trans-radial amputation with an amputation would be necessary to quantify functional improvements. Nonetheless, this study took one step toward more advanced control systems, implying that future RPH development in the direction of sensorized hands with compliant controllers would benefit people with trans-radial amputation.

6.2 Online and Dynamic Tactile Surface Exploration of Unknown Objects

In this part, we propose a novel approach for online tactile surface exploration of unknown objects while employing an adaptive compliant controller for the robotic hand. Our online exploration strategy actively and dynamically optimizes the acquired data's entropy and balances the exploration's global knowledge and local complexity. We show that this method can efficiently explore objects with arbitrary shapes, e.g., having a handle, hole, or narrow edges. To allow for multi-contact exploration with a robotic hand, we propose an optimization-based planning algorithm that adapts the hand pose to the local surface geometry and maximizes the kinematic properties of each finger online and during exploration. We benchmark our approach against the state-of-the-art in a simulated environment and showcase the strength of our method in learning object shapes with different complexities.

Probabilistic models have been widely used for representing an object's shape. Such models are employed in many applications like sensory fusion (Gerardo-Castro et al. (2015); Lee et al. (2019)) and grasp (Ottenhaus et al. (2019); Li et al. (2016a)). Probabilistic models provide information about the object surface's uncertain areas that need to be explored first. For the problem at hand, Gaussian Process Implicit Surface (GPIS) (Williams and Fitzgibbon (2006)) is used in a wealth of studies (Dragiev et al. (2011); Yi et al. (2016); Driess et al. (2017); Gandler et al. (2020)). GPIS gives the probability of points being either on, inside, or outside the object's surface; see Section 2.4.

To explore an object surface, researchers have used GPIS to either select discrete locations iteratively (Yi et al. (2016)) or query continuous paths (Driess et al. (2017)). Both approaches, however, use fixed GPIS hyperparameters. Optimizing the model parameters adds more cost to an already computationally complex model $O(N^3)$, rendering it infeasible in real-time exploration. Moreover, the optimized values of hyperparameters oscillate severely, especially in the early stages of explorations. Plenty of data points must be sampled to obtain a balanced dataset, and the likelihood of GPIS is initially far from a good metric of loss prediction w.r.t the ground truth.

Although GPIS with fixed parameters functions well for exploring simple objects, it falls short for objects with complex shapes. Loss of efficiency by visiting less informative regions and getting stuck in local minima traps are among the frequently consequent hassles. On the one hand, parameter tuning grows more tricky with the objects' variety and complexity; on the other hand, online model optimization is infeasible. Here, we separate the GP model used for path query from GP that models the implicit surface. Since exploration is only based on local information, we construct a GP model from the local data at each query point. Then we propose a dynamic approach for the evolution of parameters in the local GP used for path query. This method takes the connectivity of points and local densities and changes parameters to dynamically increase the entropy of the collected data and the exploration efficiency. This way, we avoid the issues with fixed hyperparameters and, simultaneously, acquire a more

informative and balanced sample data set from the object surface. In addition to dynamic exploration, we follow a ϵ-greedy policy we follow ϵ-greedy policy familiar to reinforcement learning (Chatzilygeroudis et al. (2020)). Occasionally allowing random exploration directions, as we will see later, helps to avoid getting stuck with absolute reliance on local information.

6.2.1 Approach

Active Path Query

Active path query in continuous exploration relies on the gradient of variance in local GP model, see Figure 6.10. The exploration needs to be as aligned as possible with the direction of gradient evolution to ensure maximum information gain. Once the inversion of $(\boldsymbol{K}+\sigma_n^2\boldsymbol{I})$ is obtained, using Cholesky factorization (Higham (1990)), we can compute the gradient at the query point:

$$\frac{\partial \mathbb{V}[f_*]}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{k}(\boldsymbol{x},\boldsymbol{x})}{\partial \boldsymbol{x}} - 2\left(\frac{\partial \boldsymbol{k}_*}{\partial \boldsymbol{x}}\right)^T (\boldsymbol{K}+\sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{k}_*. \tag{6.1}$$

On the other hand, maintaining contact with the object is essential in continuous exploration. The direction of the motion closely depends on the correct estimation of the tangent plane. We need to move along the estimated tangent plane as the object shape is unknown a priori. One approach is to extract the surface normal vector from GPIS, $\bar{f}_*$:

$$\frac{\partial \bar{f}_*}{\partial \boldsymbol{x}} = \left(\frac{\partial \boldsymbol{k}_*}{\partial \boldsymbol{x}}\right)^T (\boldsymbol{K}+\sigma_n^2\boldsymbol{I})^{-1}(\boldsymbol{y}-\boldsymbol{1}_n) \tag{6.2}$$

this gradient serves as an estimator of the tangent plane. Extracting conditional variance on a tangent plane and then taking the gradient is geometrically equivalent to projecting the gradient of the variance onto the tangent plane:

$$\boldsymbol{s} = \frac{\partial \mathbb{V}[f_*]}{\partial \boldsymbol{x}} - \frac{\left\langle \frac{\partial \mathbb{V}[f_*]}{\partial \boldsymbol{x}}, \frac{\partial \bar{f}_*}{\partial \boldsymbol{x}} \right\rangle}{\| \frac{\partial \bar{f}_*}{\partial \boldsymbol{x}} \|_2} \frac{\partial \bar{f}_*}{\partial \boldsymbol{x}} \tag{6.3}$$

from which, we are only interested in the direction $\boldsymbol{e}_s = \boldsymbol{s}/\|\boldsymbol{s}\|_2$. Instead of moving in the direction of $\boldsymbol{e}_s$, we follow ϵ-greedy policy familiar:

$$\boldsymbol{e}_s = \begin{cases} \boldsymbol{s}/\|\boldsymbol{s}\|_2 & \text{if } p \geq \epsilon \\ \boldsymbol{e}_\epsilon & \text{otherwise} \end{cases} \tag{6.4}$$

where $\epsilon \in \mathbb{R}$ is small value ($\epsilon = 0.1$ in this work). In each iteration, vector $\boldsymbol{e}_\epsilon$ is a random vector in the tangent plane at the query point, and $p \in [0\,1]$ is a randomly generated value.

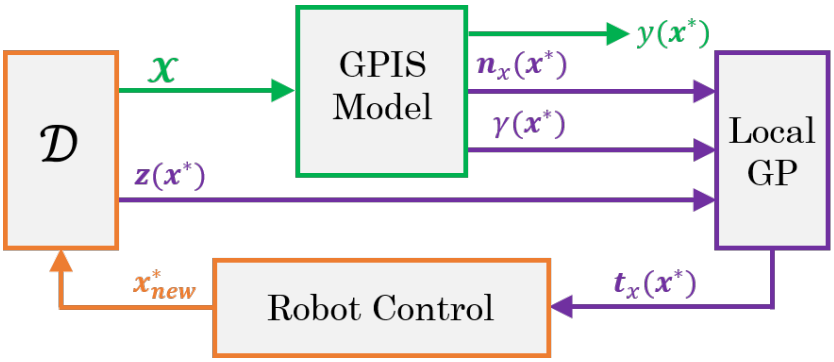


Figure 6.10: Block diagram of exploration strategy.

Dynamic Exploration

The notion of information using variance in Eq. 6.3 is local (computed from local GP), and it closely relies on the kernel hyperparameters. If the length scale ℓ, for instance, is small, then we will be exploring more the local complexities of the object, resulting in getting trapped in a local region and overloading the GP. On the contrary, we will achieve higher space coverage by considering relatively big length scales without precisely capturing the shape. Here, we endeavor to balance a trade-off between space coverage (global exploration with big length scale ℓ_b) and capturing shape complexity (local exploration with small length scale ℓ_s):

$$\ell = \omega \ell_s + (1-\omega)\ell_b \tag{6.5}$$

with $\omega \in [0\,1]$ being the balancing weight. In order to determine the value of ω we define two metrics here, one for local density and the other for information gain. Let's define $\rho \in [0\,1]$ as a measure of local density which we compute for a query point $\boldsymbol{x}_*$ via :

$$\rho(\boldsymbol{x}_*) = \frac{\mathbb{E}[\boldsymbol{z}^2] - \mathbb{E}[\boldsymbol{z}]^2}{\rho^2_{max}} \tag{6.6}$$

where $\boldsymbol{z} = \{\boldsymbol{z} | \boldsymbol{z} \in \text{k-}NN(\boldsymbol{X}, \boldsymbol{x}_*)_P\}$ represents recorded points on the object surface and within the subset of P nearest neighbors to the query points. The set bound value ρ_{max} is the defined maximum effective density. Big values of ρ correspond to the fact that P-nearest points are widely spread, while small values represent a dense neighborhood. We also consider the connectivity of the query point in GP to measure the information gain. The connectivity γ for a query point is computed thorough:

$$\gamma(\boldsymbol{x}_*) = \mathbf{1}_M^T \boldsymbol{k}_* / M \tag{6.7}$$

and M is the dimension of $\boldsymbol{k}_*$ or the number of all points presented to GP. The growth in value of γ means that newly acquired data are similar to those already collected, representing a

Algorithm 4 Exploration strategy for a single query point

Parameters: ℓ_s, ℓ_b, and ℓ_0
Initialization:
$\boldsymbol{x} \leftarrow$ random ▷ initialize query point randomly on object
$\mathscr{X} \leftarrow \{\boldsymbol{x}\}$ ▷ initial dataset
$\mathscr{Y} \leftarrow \{0\}$ ▷ initial dataset
$\omega \leftarrow 0$, $\gamma \leftarrow 1$ ▷ initialize balancing weight and connectivity measure
Main loop:
for $i = 1 \rightarrow T$ **do**
  $\boldsymbol{x}_* \leftarrow \mathscr{X}(end)$
  $y_* \leftarrow \mathscr{Y}(end)$
  update $(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})$ ▷ with new ℓ
  re-compute $(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}$
  $n_x \leftarrow \frac{\partial \bar{f}_*}{\partial \boldsymbol{x}}$ ▷ normal of the surface
  $t_x \leftarrow \frac{\partial \mathbb{V}[f_*]}{\partial \boldsymbol{x}}$ ▷ variance gradient
  $\{\boldsymbol{x}_{new}, y_{new}\} \leftarrow$ path query $\leftarrow n_x$ and t_x
  $\mathscr{X} \leftarrow \boldsymbol{x}_{new}$ and $\mathscr{Y} \leftarrow y_{new}$
  $m_P \leftarrow min(P, size(\mathscr{X}))$
  $z \leftarrow KNN_{m_P}(\boldsymbol{x}_*, \mathscr{X})$
  $\rho(\boldsymbol{x}_*) \leftarrow \frac{\mathbb{E}[z^2] - \mathbb{E}[z]^2}{\rho^2_{max}}$
  $\gamma(\boldsymbol{x}_*) \leftarrow \mathbf{1}_M^T \boldsymbol{k}_* / M$
  $\dot{\omega} \leftarrow -\beta \, \text{Relu}(\dot{\gamma})(\omega - \rho)$
  $\omega \leftarrow \omega + \dot{\omega}\delta$
  $\ell \leftarrow \omega \ell_s + (1 - \omega)\ell_b$
end for

decrease in information gain. Given these, propose the following dynamic for ω:

$$\dot{\omega} = -\beta \, \text{Relu}(\dot{\gamma})(\omega - \rho) \tag{6.8}$$

in which, $\beta \in \mathbb{R}^+$ gauges the convergence rate. Based on Eq. 6.8 and 6.5, the value of ℓ alters to avoid dense locals (small ρ) by more globally exploring (bigger ℓ) and to capture shape complexities (smaller ℓ) when the global coverage is fine (big ρ). Moreover, the value of ℓ changes only when information gain decreases or γ increases. This ensures that any model modification is executed to improve the entropy of the collected data. Algorithm 4 summarizes the exploration strategy for a given query point.

Robotic Hand Control

The state-of-the-art robot exploration strategies of unknown objects are rather restricted. In most cases, the entire robotic system is considered one single agent. For example, a robotic arm uses its end-effector mounted with tactile sensors to explore an unknown workspace and objects (Kaboli et al. (2017)); or a bimanual robotic system with both robotic hands to hold

and explore the geometric shape of an unknown object (Sommer et al. (2014)). In this scenario, the multiple fingers of the robotic hand are only used to wrap the object surface, despite the abundant degrees of freedom in each finger.

In comparison, when exploring the geometric shape of an unknown object relying on the sense of touch, humans usually first use fingers to establish multiple contacts with the object and then move hand and fingers to travel through different object surface regions. During this process, the hand and fingers move in compliance to adapt to the object's surface's geometry. Moreover, although all fingers move together with the hand, they differ slightly in individual movements, such that the surface features of the local regions can also be acquired. This combination of hand (i.e., *global exploration*) and individual finger movements (i.e., *local exploration*) affords an efficient exploration of unknown objects.

To allow for multi-contact exploration with a robotic hand, we use an optimization-based planning algorithm that adapts the hand pose to the local surface geometry online, and maximizes the kinematic properties of each finger during exploration. This method is developed by Kunpeng Yao and presented in Appendix C.2. In particular, this approach allows online regulation of hand pose, such that the hand pose adapts to the regional curvature of the explored surface, and each finger also moves with higher flexibility, in the sense that each finger's configuration is far from its kinematic singularity, while can potentially explore the larger local region. Moreover, we consider constraints of the robotic system by formulating them as problem constraints, such that physically feasible and collision-free hand poses can be obtained.

6.2.2 Experimental Evaluation

We have evaluated our proposed approaches in simulation. In our experiments, the task is performing tactile exploration of unknown objects. We use various objects with different levels of shape complexities, and ground truths (objects point cloud) are used only to evaluate the method's performance in reconstructing the object shape. We devise the following scenarios for experimental evaluation of our method:

(i) we assess, in Section 6.2.2, the effectiveness of our contributions by comparing each of them with the state-of-the-art method; and

(ii) in Section C.2.3, we validate our approach in simulation for tactile exploration of various objects.

Comparison with State of the Art

We benchmark the contributions of our proposed dynamic exploration strategy against the state-of-the-art methods in following tests.

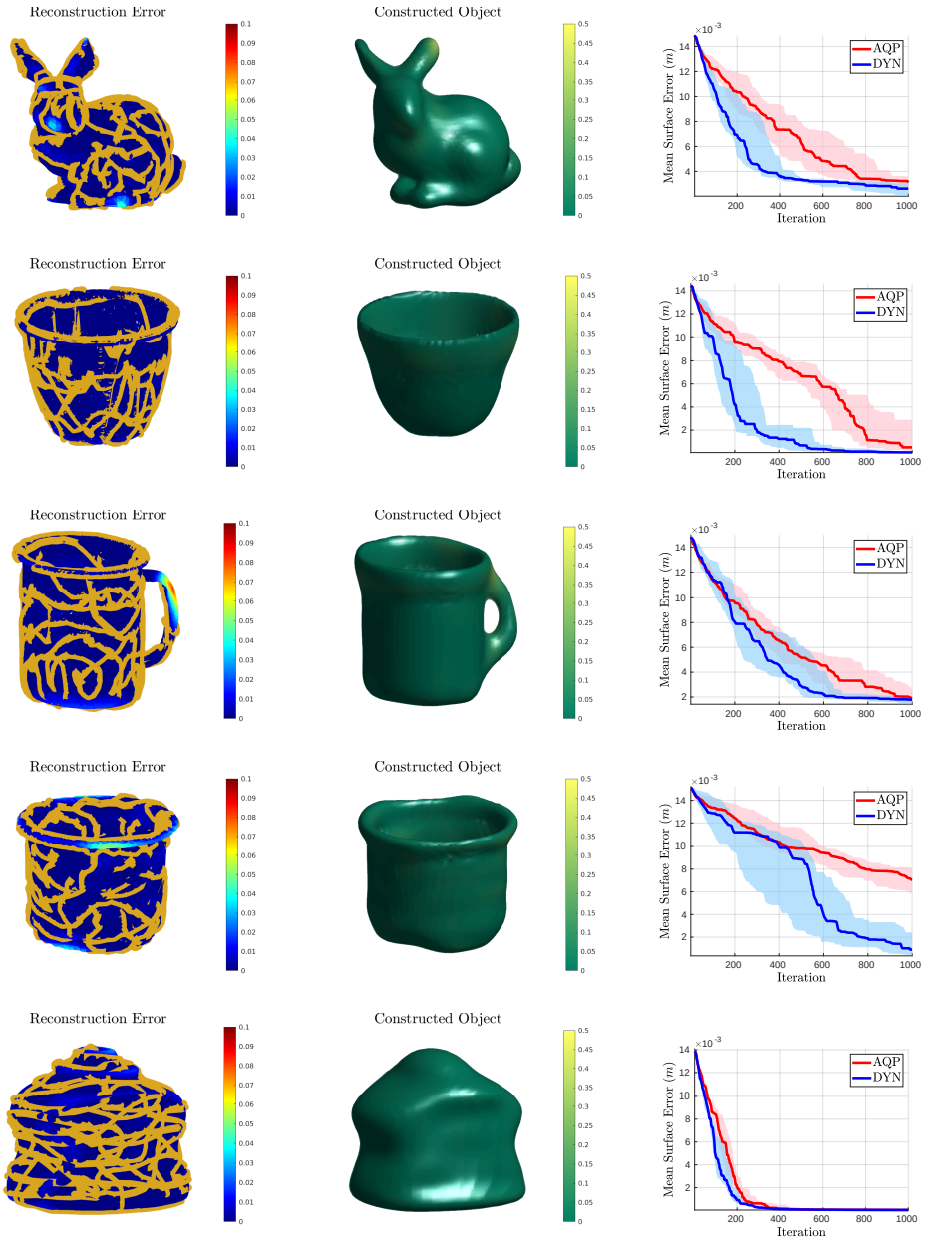


Figure 6.11: Examples of shape reconstruction via online exploration. Objects are unknown a priori. The top row shows the reconstruction error and the acquired data along the traversed trajectory on the object surface, using our exploration method. The middle row demonstrates the the object shape predicted by the learned GPIS model. The bottom row represents the exploration performance in terms of the mean surface error using our approach (DYN) and the state-of-the-art (AQP) (Driess et al. (2019)). For 10 replicates and in each run, the initial point is set randomly, and remains the same for both methods. The solid line indicates the median and shaded area is 25 to 75 percentile of these 10 replicates.

In a single contact exploration where only one tactile probe is in contact with the object, we compare the exploration performance of our dynamic exploration (Section 6.2.1) and the state-of-the-art active query (Driess et al. (2019)). This experiment aims to assess the method's effectiveness in planning more informative trajectories and collecting more efficient data. For this reason, we relax all robotic constraints by assuming a single and free tactile prob that can move continuously on the object's surface.

Figure 6.11 illustrates the reconstruction error, traversed trajectory, shape prediction, and method comparison for this experiment. From Figure 6.11 we find that both methods are identically perfect for a convex and simple shape object (the last object from left in Figure 6.11). However, for other objects, our approach collects points with significantly higher information values, hence lower reconstruction error. We constantly minimize the connectivity of points in our method to obtain less similar points. Balancing local and global exploration helps us to avoid local traps. For instance, imagine one of the objects like a bowl, cup, or glass (second, third, and fourth objects in Figure 6.11). We are hardly, if not never, able to enter/exit the interior part of the object using the classical method. However, balancing local density forces exploration to improve the overall knowledge of the object shape before getting trapped by showing interest in complex local regions. We occasionally allow random directions of exploration. This helps us to avoid absolute reliance on the acquired data from a completely unknown object; in this way, it is more likely to find the ears of the bunny rabbit (the first object from left in Figure 6.11).

Hand and Fingers Pose Adaptation

In a multi-contact exploration scenario with a 16 DoFs robotic hand, we compare our hand pose adaptation algorithm with the baseline (Sommer and Billard (2016)). Here, the tactile probes are no longer free/independent agents (on contrary to (Driess et al. (2019))) and they need to physically feasible trajectories w.r.t robotic kinematic constraints.

In (Sommer and Billard (2016)) the robotic hand tries to maximize the number of contact points with the object, and while the hand (the wrist, the base of the robotic hand) is moving in predefined trajectories, only the contact point with information value are appended to the data. In our approach, however, we plan trajectories automatically and in an informative way; i.e., we adapt both the hand pose and finger joints to query most informative path for the entire robotic hand to move.

Since the efficacy of the baseline (Sommer and Billard (2016)) depends on the predefined trajectory for the hand wrist, to have a fair comparison, we use the same technique as the one in our approach to extract the wrist position trajectory. Thus, the only difference among methods is that (Sommer and Billard (2016)) maximizes the number of contacts as the hand moves whereas we adapt joint position and the wrist orientation to query informative contact points. Figure 6.12 summarizes the performance of both approaches when exploring different objects. From Figure 6.12 we observe that both methods work similarly perfect when exploring

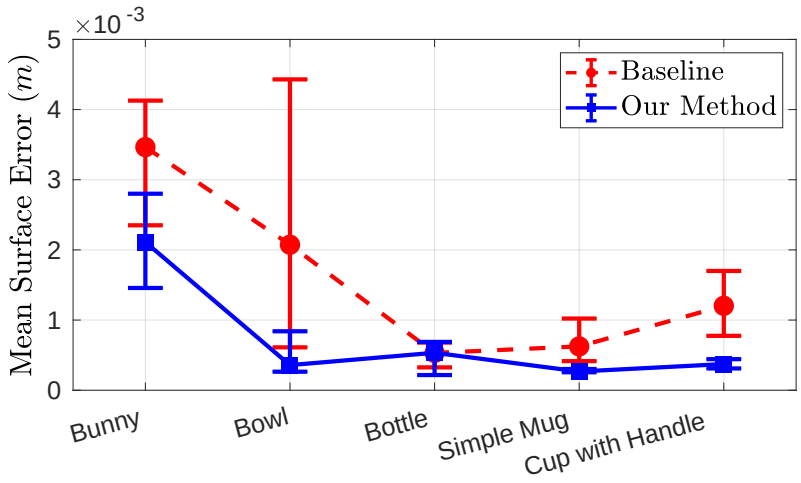


Figure 6.12: Performance comparison of our exploration method (solid line) with Multi-Contact method of Sommer and Billard (2016)(dash line). Markers, lower caps and upper caps indicate, respectively, the medians, 25% and 75% percentiles over 20 replicates of the simulation.

simple/convex objects. However, our method is significantly more accurate and consistent once the unknown objects are with shape complexities (e.g., the handle of the mug).

6.2.3 Discussion and Summary

Object shape estimation through autonomous tactile exploration is a fundamentally multi-faceted problem and yet essential in robotics. The challenges amplify if the object's geometry is a priori unknown, as complex shapes may hinder the exploration of such objects.

In this study, we investigated the problem of exploring the geometry of an unknown target object using a multi-fingered robotic hand with tactile sensors. We proposed algorithms to tackle two main challenges in the related field: namely (i) a GPR-based path generation algorithm to balance the global and local information for multiple agents and (ii) an adaptation algorithm to improve the movement of the robotic hand by leveraging both the inherent model property and obtained local information.

We start with reasoning about the GPIS model selection suitable for robotic exploration. We then proposed a method that dynamically alters hyperparameters of GPIS to increase the information gain (entropy) of collected data online. We planned for active balancing of local and global queries. Our online exploration algorithm tries to train a GPIS model by obtaining a global understanding of the object's shape before focusing on local complexities. Using various objects, we benchmark against the state-of-the-art. We showed that our online exploration algorithm provides higher data efficiency in exploring surfaces of complex objects.

We focused on acquiring a planning algorithm that functions in a complex object exploration

and can be applied to multi-finger robotic hands. Optimizing the GPIS model online is not only infeasible but also inconsistent with the problem at hand. Computational cost, early value oscillation, and an unbalanced dataset are among the reasons. Instead, we propose to dynamically modulate GPIS based on the connectivity of points and local densities. We verified that in the isolated exploration problem (e.i. no robotic constraints included), our method is robust to data efficiency loss. We actively react to visiting less informative regions, and by balancing local and global exploration, we avoid getting stuck in local minima traps. We showcased the strength of this method by benchmarking against state-of-the-art in exploring objects with various shapes.

Moreover, in most state-of-the-art studies, a strategy for dynamically adapting the hand pose during exploration is absent. This may lead to a collision between the robotic hand and the target object and undermine the finger's motion ability in exploration, such as getting stuck in a kinematic singular configuration or closing to its joint limit. Our proposed algorithm tackles these two issues by integrating a dynamic hand pose regulation strategy to adapt the hand orientation to the local surface geometry, together with a finger kinematics optimization strategy to prevent fingers from singular configurations and increase the potential reachable regions of a fingertip. Our simulation results demonstrated the effectiveness of our proposed strategy in both aspects. The robotic hand was able to explore the unknown surface with as much collision as possible.

Despite the successful application, there are still a couple of limitations and room for improvement. We assumed that the object was fixed and that the robotic hand could safely interact with the object. This, however, is rarely the case in practice. Object displacement is inevitable in object-hand interactions. Tracking and estimating object pose and stabilizing it in hand is necessary in such cases. We, humans, amazingly explore objects while manipulating them in hand and, at the same time, improve our estimation of the shape. Reaching a human-level exploration performance is our ultimate objective, toward which we took only a single step in this study.

In addition, although our algorithm can largely improve the kinematic properties of a robotic hand during exploration by leveraging the information of explored regions, it does not guarantee a fully collision-free trajectory. This is mainly due to the limited prior knowledge of the target object. Collision is inevitable for objects with complex shapes, such as non-continuous surfaces or non-convex geometry. It is theoretically infeasible to avoid potential collision in an unknown space actively. This limitation can be alleviated by introducing extra prior knowledge into the problem formulation, such as having a more specific description of the object's geometry.

7 Conclusions

This chapter reviews the research conducted throughout the thesis and summarizes the contributions. Then, we describe the limitations and highlight the potential directions for future work.

7.1 Contributions

This thesis took inspiration from humans' mastery to (i) remain highly compliant and quickly respond to uncertainties in an unstructured environment and (ii) learn, adapt, and master new dexterous tasks. The central contribution of the thesis is developing a robot controller for dexterous grasp and manipulation compliant in interaction and faithful in tracking. We first focused on devising methods that can constantly update and improve the available models of dynamics and can adapt and react to uncertainties online. Next, we introduced a coordinated multi-finger system to our controller. We showed that this solution provides sufficient robustness for grasping and manipulating problems during task execution in uncertain environments.

We first reviewed the necessary technical background of the thesis. Then, we dedicated the third chapter to the challenges of learning the Inverse Dynamics (ID) model of a robot manipulator. The ID model is instrumental in precise robot control and a key component in compliant manipulation. We investigated supervised machine learning techniques to incrementally explore a robot's configuration space and maximize the information of the collected data. We also introduced new excitation trajectories that impose stable limit cycles in robot joints' phase space while satisfying feasibility constraints and physical bounds. We showed that the data collected by our approach is significantly richer than the collected data with traditional techniques. Since, in any learning framework, the efficacy of the learned model substantially depends on the training data supplied to it, our method has resulted in better learning performance than the state-of-the-art methods, regardless of the learning method.

Chapter 4 proposed a novel combination of a model reference adaptive controller with a residual ID model augmented Quadratic Programming (QP) based controller. The adaptive control law adjusts the QP online to account for model uncertainties and unforeseen disturbances, whereas the residual model learning episodically captures the unmodeled dynamics. We extensively validated our approach, called Self-Correcting Quadratic Programming (SCQP), in simulations and performed experiments in a physical robotic setup. We show that SCQP compensates, in a handful of trials, for significant unmodeled dynamics in various tasks ranging from simple end-effector tracking to humanoid balancing and robotic hand object grasping. Also, SCQP helps us to avoid the tedious tuning of PID controllers despite the task variability.

Chapter 5 addressed the problem of ensuring robust dexterous manipulation when facing a poor model of the object's dynamics, model imperfections and external disturbances. In this chapter, we proposed a novel method to coordinate robotic fingers based on dynamical systems. Our coupled DS uses an intermediate dynamic to synchronize all fingers. This method offers a robust and coordinated multi-finger system for various grasp and manipulation experiments with different objects. We combined our adaptive controller with joints' impedance regulation to guarantee high tracking accuracy and adapt to dynamic changes. Our experiments on a 16-DoFs robotic hand showed that the controller could compensate for its dynamics and stably manipulates objects with different mass properties. We also showcased that our controller, combined with learning from human demonstration, provides a robust solution for more complex dexterous tasks such as finger gaiting.

In Chapter 6, we use our robotic hand controller in two problems and demonstrate its applicability in human-centric environments. In the first application, we aimed to increase the dexterity of robotic prosthetic hands (RPHs) for patients with a hand amputation. We studied four shared-control conditions based on a compliant controller combined with EMG decoding in order to teleoperate a robotic arm, maintaining complete autonomy over high-level commands. We evaluated our shared-control approach in an experimental setup by conducting a 3-days long longitudinal study with healthy individuals. Results indicated that when combined with incremental EMG decoding, robotic assistance leads to significantly less failure and faster completion time in task execution. Thus, it could be a realistic alternative for users with trans-radial amputation to improve the dexterity and versatility of RPHs.

Finally, in the second application of Chapter 6, we investigated the problem of exploring the geometry of an unknown target object using a multi-fingered robotic hand with tactile sensors. We proposed algorithms to tackle two main challenges in the related field: (i) a Gaussian Process-based path generation algorithm to balance the global and local information for multiple agents and (ii) an adaptation algorithm to improve the movement of the robotic hand by leveraging on both the inherent model property and obtained local information. Using various objects, we benchmarked our method against the state-of-the-art. We showed that our online exploration algorithm provides higher efficiency in exploring surfaces of complex objects.

7.2 Limitations and Future Work

The work presented in the thesis has limitations that open doors to future research and investigation.

In many applications, the robot should work in a predefined workspace. Acquiring an accurate ID model outside this workspace is not necessary. Therefore, performing the exploration with our approach in Chapter 3, Max-Information Configuration Exploration (MICE), only within the desired workspace in the configuration could be used to improve the efficiency of the data further. Moreover, using incremental model learning approaches (Nguyen-Tuong and Peters (2010)) in conjunction with MICE is another potential topic of study which can improve the efficiency of model learning.

In Chapter 4, we assumed that the learned ID model could generalize to the underlying dynamics. In some cases, however, this assumption is not realistic enough. For instance, in the bimanual grasping task, Section 4.4.3, each contact model might require a different contact configuration, and using one model for different contacts might not be comprehensive. Representing an ID model that can explain different contact (Calandra et al. (2015)) can help the residual model learning to generalize better to the dynamics of the systems. Also, the high-level tasks remained fixed throughout the experiments of this chapter. Task modulating can be crucial if the original specifications are not achievable; for example, with a humanoid, the left arm is damaged, and the task has to be performed with the right arm. Integrating SCQP with methods that update the high-level planning can be helpful in such scenarios. Moreover, our approach can be combined with reinforcement learning algorithms. One idea is to enable exploration around the commands the QP outputs while ensuring safety and not allowing deviations that could potentially harm the robot.

In Chapter 5, we employed tactile sensors at the fingertips to obtain forces at contact points and used a motion capture system to estimate object position. However, humans benefit from rich and advanced sensory feedback when holding an object in hand. Tactile sensing in the entire hand surface (e.g., in finger bodies) enables humans to detect not only the pose of the object but also how securely it is grasped. Increasing the efficiency and comprehensiveness of perception in robotic hands would be a huge step toward achieving a more human-like dexterity. For instance, when estimating the object's pose within the hand by the OptiTrack system, markers occlusion during fingers motion occurs frequently. Loss of tracking of the object can endanger grasp stability. In this case, being able to perceive and track object position or the net applied force through tactile feedback will enhance the reliability of the perception. Thus, rich sensory feedback can significantly increase the control's accuracy and robustness. As another example, slippage, especially for heavier objects, caused some experiment trials to fail. Slippage detection and appropriate adaptation could improve the success rate.

For the first application in Chapter 6, integrating a robotic prosthetic hand (RPH) and validation on people with trans-radial amputation would be necessary to quantify functional

improvements. Another step to increase intuitiveness and sense of agency for users is to robustify the EMG decoder commands and address the signals' variability issue. Moreover, sensory feedback can help users to send more accurate commands to the robotic hands, improve embodiment (Bensmaia et al. (2020)) and reduce cognitive load (Valle et al. (2020)). The estimation of the object's pose relative to the robotic hand was only based on the forward kinematic and the fingers' joint position. This estimation proved to be helpful with a simple cuboid object; however, for more complex object shapes, other estimation methods like vision (Doosti et al. (2020)) or tactile images Sodhi et al. (2021)) are required.

In the second application of Chapter 6, we assumed that the object was fixed and that the robotic hand could safely interact with the object. Humans usually explore objects while manipulating them in hand. In this case, the object is not fixed. Exploring objects by manipulating them could make the exploration strategy more efficient. However, tracking and estimating object pose and stabilizing it within the robotic hand during exploration is a difficult task, which remains an open problem that deserves further investigation.

A Appendix of Chapter 4

A.1 Adaptive Control Stability Proof

Let $\mathbf{e} = \boldsymbol{\zeta} - \boldsymbol{\zeta}_r$ be the tracking error. Using Eq. 4.5, 4.6 and 4.7, the error's dynamics (time derivative) is given by:

$$\dot{\boldsymbol{e}} = (\mathbf{A} + \mathbf{B}\Psi_\zeta)\boldsymbol{\zeta} - \mathbf{A}_r\boldsymbol{\zeta}_r + (\mathbf{B}\Psi_r - \mathbf{B}_r)\boldsymbol{r}(t) + \mathbf{B}\Psi_\phi\boldsymbol{\Phi}(\boldsymbol{e}) + \mathrm{F}(\boldsymbol{\zeta}_d, \boldsymbol{\zeta}). \tag{A.1}$$

For the error dynamic (A.1) to follow the reference dynamic (4.5), we set that the desired gain matrices Ψ^*_ζ, Ψ^*_r, and Ψ^*_ϕ satisfy:

$$\begin{aligned} \mathbf{A} + \mathbf{B}\Psi^*_\zeta &= \mathbf{A}_r \\ \mathbf{B}\Psi^*_r &= \mathbf{B}_r \\ \mathbf{B}\Psi^*_\phi\boldsymbol{\Phi}(\boldsymbol{e}) &= -\mathrm{F}(\boldsymbol{\zeta}_d, \boldsymbol{\zeta}). \end{aligned} \tag{A.2}$$

Given that system (4.5) is by construction stable (Matrices $\mathbf{P}, \mathbf{A}_r$ satisfy $\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$; see Section 4.3.1), then $\boldsymbol{e} \to 0$. We define the gain prediction errors as $\widetilde{\Psi}_\zeta = \Psi_\zeta - \Psi^*_\zeta$, $\widetilde{\Psi}_r = \Psi_r - \Psi^*_r$, and $\widetilde{\Psi}_\phi = \Psi_\phi - \Psi^*_\phi$. Replacing (A.2) in (A.1), we obtain:

$$\dot{\boldsymbol{e}} = \mathbf{A}_r\boldsymbol{e} + \mathbf{B}\widetilde{\Psi}_\zeta\boldsymbol{\zeta} + \mathbf{B}\widetilde{\Psi}_r\boldsymbol{r}(t) + \mathbf{B}\widetilde{\Psi}_\phi\boldsymbol{\Phi}(\boldsymbol{e}) \tag{A.3}$$

Theorem 1. *The error dynamics (A.3) is Lyapunov stable, and the tracking error, $\boldsymbol{e}$, vanishes asymptotically under the proposed control law (4.7) and adaptation law (4.10).*

Proof. Consider the following Lyapunov function:

$$\mathbf{V}(\boldsymbol{e}, \widetilde{\Psi}_\zeta, \widetilde{\Psi}_r, \widetilde{\Psi}_\phi) = \frac{1}{2}\boldsymbol{e}^T\mathbf{P}\boldsymbol{e} + \frac{1}{2}\mathbf{tr}\big(\widetilde{\Psi}_\zeta^T\Theta_\zeta\widetilde{\Psi}_\zeta + \widetilde{\Psi}_r^T\Theta_r\widetilde{\Psi}_r + \widetilde{\Psi}_\phi^T\Theta_\phi\widetilde{\Psi}_\phi\big) \tag{A.4}$$

where P, Θ_ζ, Θ_r, and $\Theta_\phi \succ 0$. V is positive. It is zero when both the tracking error vanishes and

the gains no longer vary. Taking the time derivative of Eq.A.4 yields:

$$\dot{\mathbf{V}} = \frac{1}{2}\boldsymbol{e}^T(\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P})\boldsymbol{e} + \boldsymbol{e}^T\mathbf{P}\mathbf{B}\big(\widetilde{\Psi}_\zeta\boldsymbol{\zeta} + \widetilde{\Psi}_r\boldsymbol{r}(t) + \widetilde{\Psi}_\phi\boldsymbol{\Phi}(\boldsymbol{e})\big) + \mathbf{tr}\big(\widetilde{\Psi}_\zeta^T\Theta_\zeta\dot{\widetilde{\Psi}}_\zeta + \widetilde{\Psi}_r^T\Theta_r\dot{\widetilde{\Psi}}_r + \widetilde{\Psi}_\phi^T\Theta_\phi\dot{\widetilde{\Psi}}_\phi\big). \tag{A.5}$$

From $\mathbf{B}\Psi_r^* = \mathbf{B}_r$, we replace $\mathbf{B} = \mathbf{B}_r\Psi_r^{*-1}$ (if $\mathbf{B}_r \succ 0$, then $\Psi_r \succ 0$ and inevitable by construction (Culbertson and Schwager (2018))). Setting $\Theta_\zeta = \Psi_r^{*-T}\Lambda_\zeta^{-1}$, $\Theta_r = \Psi_r^{*-T}\Lambda_r^{-1}$, and $\Theta_\phi = \Psi_r^{*-T}\Lambda_\phi^{-1}$, we can rewrite Eq.A.5 as:

$$\dot{\mathbf{V}} = -\frac{1}{2}\boldsymbol{e}^T\mathbf{Q}\boldsymbol{e} + \boldsymbol{e}^T\mathbf{P}\mathbf{B}_r\Psi_r^{*-1}\big(\widetilde{\Psi}_\zeta\boldsymbol{\zeta} + \widetilde{\Psi}_r\boldsymbol{r}(t) + \widetilde{\Psi}_\phi\boldsymbol{\Phi}(\boldsymbol{e})\big) + \mathbf{tr}\big(\widetilde{\Psi}_\zeta^T\Psi_r^{*-T}\Lambda_\zeta^{-1}\dot{\widetilde{\Psi}}_\zeta + \widetilde{\Psi}_r^T\Psi_r^{*-T}\Lambda_r^{-1}\dot{\widetilde{\Psi}}_r\widetilde{\Psi}_\phi^T\Psi_r^{*-T}\Lambda_\phi^{-1}\dot{\widetilde{\Psi}}_\phi\big) \tag{A.6}$$

where matrices Λ_ζ, Λ_r, and Λ_ϕ are positive definite, and tune the convergence rate of the adaptive gains. Replacing the adaptation law Eq. (4.10) in Eq. (A.6), and taking advantage of the trace property for square matrices $\mathbf{tr}(\mathbf{AB}) = \mathbf{tr}(\mathbf{BA})$, all the right hand side terms of Eq.A.6, except the first one, vanish and the Lyapunov time derivative simplifies into: $\dot{\mathbf{V}} = -\frac{1}{2}\boldsymbol{e}^T\mathbf{Q}\boldsymbol{e}$.

Since $\mathbf{V} > 0$ and $\dot{\mathbf{V}} \leq 0$ the system in Eq. (A.3) is Lyapunov stable, thus $\boldsymbol{e}, \widetilde{\Psi}_\zeta, \widetilde{\Psi}_r$, and $\widetilde{\Psi}_\phi$ are bounded. As $\boldsymbol{r}(t)$ is bounded (see Section 4.3.1), the system given by Eq. (A.3) and all variables in closed-loop remain bounded. By extension, $\ddot{\mathbf{V}} = -\boldsymbol{e}^T\mathbf{Q}\dot{\boldsymbol{e}}$ remain bounded at all time. Thus, from Barbalat's lemma (Khalil and Grizzle (2002)) $\dot{\mathbf{V}} \to 0$ with $t \to \infty$ and $\boldsymbol{e} \to 0$.

A.2 Function Approximation for Adaptive Control

In this section, the adaptive control algorithm assumes that system's nonlinearities are encapsulated in a nonlinear function F(·) which is need to be approximated online. One way to approximate F(·) is using universal function approximators. Neural Networks (NN), mostly in the form of RBF neural networks, are widely adopted (Seshagiri and Khalil (2000)). One reason of their popularity is that RBFNs mathematically represent a class of linear-in-the-weight approximators (Chowdhary et al. (2014)). F(·) can be approximated by $\mathrm{F}(\cdot) = \Psi_\phi^*\boldsymbol{\Phi}(\cdot) + \epsilon^*$ where ϵ^* denotes the network reconstruction error. With $\psi_i \in \mathbb{R}^n$ being the NN weight for the ith node, we can construct the weight matrix $\Psi_\phi^* = [\psi_1 \quad \psi_2 \quad \dots \quad \psi_p]$. Also $\boldsymbol{\Phi}(\cdot) = [\phi_1(\cdot) \quad \phi_2(\cdot) \quad \dots \quad \phi_p(\cdot)]^T$ is a vector of radial basis functions in which $\phi_i(\cdot)$ the ith node function is given by $\phi_i(\mathrm{x}) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{\|\mathrm{x}-\boldsymbol{\xi}_i\|^2}{2\sigma_i^2}\right)$, with $\boldsymbol{\xi}_i \in \mathbb{R}^n$ the center, and σ_i the corresponding kernel width for RBF kernel of the ith node. It is noteworthy that basis function $\phi_i(\cdot)$ is not restricted to RBF kernels and one could select other types of kernels such as Cosine, or Polynomial kernels. We adopt this methodology to estimated unmodeled nonlinearities in multi-body systems, described in Section 4.3.1.

B Appendix of Chapter 5

B.1 Joint-Space Control

B.1.1 Computing the Regulation Signal

We can use any inverse kinematic (IK) solver (Feng et al. (2014)) or more advanced probabilistic IK models (Li et al. (2016a)) for mapping the desired velocity (Eq. 5.9) to the desired state $\boldsymbol{\zeta}_i^d$, then we can use $\boldsymbol{r} = -\mathbf{B}_r^{\dagger}\mathbf{A}_r\boldsymbol{\zeta}_i^d$ to compute the regulation signal ($\mathbf{B}_r^{\dagger}$ is the pseudo inverse of $\mathbf{B}_r$). Although this computation coordinates well with the rest of our control derivation, we offer the following quadratic optimization:

$$\min_{\boldsymbol{r},\boldsymbol{q}_i',\dot{\boldsymbol{q}}_i'} -\frac{\mathrm{w}_1}{2}\|\boldsymbol{\delta}\|^2 - \frac{\mathrm{w}_2}{2}\|\boldsymbol{q}_i'\|^2$$
$$\text{s.t.} \begin{cases} \mathbf{B}_r\boldsymbol{r} + \mathbf{A}_r[\boldsymbol{q}_i'^T, \dot{\boldsymbol{q}}_i'^T]^T = 0 \\ \mathbf{J}\dot{\boldsymbol{q}}_i' = \dot{\boldsymbol{x}}_i^d + \boldsymbol{\delta} \\ \boldsymbol{q}_i' - \dot{\boldsymbol{q}}_i'\,dt = \boldsymbol{q}_i \\ \boldsymbol{q}_i' \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}] \\ \dot{\boldsymbol{q}}_i' \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}] \end{cases} \tag{B.1}$$

with $\boldsymbol{\zeta}_i^d = [\boldsymbol{q}_i'^T, \dot{\boldsymbol{q}}_i'^T]^T$, and weights $\mathrm{w}_1 \in \mathbb{R}$, and $\mathrm{w}_2 \in \mathbb{R}$ that tune the importance of reaching the desired velocity and penalizing non-uniform joint angles. This optimization scheme will be solved at every time step and works similar to other IK solvers; however, it favors a more uniform joint configuration and smoother velocity transition, suitable for the problem at hand. It should be noted that we assumed the solution of IK for a given $\dot{\boldsymbol{x}}_i^d$ exists, which relies on appropriate task planning and hardware limits. The stability of DS (5.9) and IK optimization in (B.1) guarantee that $\boldsymbol{r}$ remains bounded.

B.1.2 Stability Proof of Adaptive Control

We define $\mathbf{e} = \boldsymbol{\zeta} - \boldsymbol{\zeta}_i^d$ to be the tracking error. From Eq. 5.15 and Eq. 5.16 with the controller in Eq. 5.17, we can construct the error dynamics as

$$\dot{\boldsymbol{e}} = (\mathbf{A} + \mathbf{B}\Psi_\zeta)\boldsymbol{\zeta} - \mathbf{A}_r\boldsymbol{\zeta}_r + (\mathbf{B}\Psi_r - \mathbf{B}_r)\boldsymbol{r})$$

and if we assume that there exist Ψ_ζ^*,and Ψ_r^* such that

$$\begin{aligned} \mathbf{A} + \mathbf{B}\Psi_\zeta^* &= \mathbf{A}_r \\ \mathbf{B}\Psi_r^* &= \mathbf{B}_r \end{aligned} \tag{B.2}$$

then the error dynamics will be summarized as

$$\dot{\boldsymbol{e}} = \mathbf{A}_r\boldsymbol{e} + \mathbf{B}\widetilde{\Psi}_\zeta\boldsymbol{\zeta} + \mathbf{B}\widetilde{\Psi}_r\boldsymbol{r} \tag{B.3}$$

where $\widetilde{\Psi}_\zeta = \Psi_\zeta - \Psi_\zeta^*$ and $\widetilde{\Psi}_\mathrm{r} = \Psi_\mathrm{r} - \Psi_\mathrm{r}^*$ are gain prediction errors. The goal is to have adaptation laws that stabilize this system and regulate prediction errors. For this, we consider the following Lyapunov function:

$$\mathbf{V}(\boldsymbol{e}, \widetilde{\Psi}_\zeta, \widetilde{\Psi}_r) = \frac{1}{2}\boldsymbol{e}^T\mathbf{P}\boldsymbol{e} + \frac{1}{2}\mathbf{tr}\big(\widetilde{\Psi}_\zeta^T\Theta_\zeta\widetilde{\Psi}_\zeta + \widetilde{\Psi}_r^T\Theta_r\widetilde{\Psi}_r\big)$$

where Θ_ζ and $\Theta_r \succ 0$. Differentiating in time, we have:

$$\begin{aligned} \dot{\mathbf{V}} = &\frac{1}{2}\boldsymbol{e}^T(\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P})\boldsymbol{e} + \boldsymbol{e}^T\mathbf{P}\mathbf{B}\big(\widetilde{\Psi}_\zeta\boldsymbol{\zeta} + \widetilde{\Psi}_r\boldsymbol{r}\big) \\ &+ \mathbf{tr}\big(\widetilde{\Psi}_\zeta^T\Theta_\zeta\dot{\widetilde{\Psi}}_\zeta + \widetilde{\Psi}_r^T\Theta_r\dot{\widetilde{\Psi}}_r\big) \end{aligned} \tag{B.4}$$

To make the Lyapunov derivative independent of the unknown matrix $\mathbf{B}$, from Eq. B.2, we assume $\mathbf{B} = \mathbf{B}_r\Psi_r^{*-1}$, for which the existence of Ψ_r^{*-1} relies on $\mathbf{B}$ being full rank or the system (Eq. 5.16) being controllable. As we place the adaptive control on joint torques, the system (Eq. 5.16) is fully controllable. Thus, this assumption is not restrictive for our controller. Given this, we can replace Θ_ζ and Θ_r with:

$$\begin{aligned} \Theta_\zeta &= \Psi_r^{*-T}\Lambda_\zeta^{-1} \\ \Theta_r &= \Psi_r^{*-T}\Lambda_r^{-1} \end{aligned}$$

that converts Eq. B.4 to:

$$\begin{aligned} \dot{\mathbf{V}} = &-\frac{1}{2}\boldsymbol{e}^T\mathbf{Q}\boldsymbol{e} + \boldsymbol{e}^T\mathbf{P}\mathbf{B}_r\Psi_r^{*-1}\big(\widetilde{\Psi}_\zeta\boldsymbol{\zeta} + \widetilde{\Psi}_r\boldsymbol{r}_t\big) \\ &+ \mathbf{tr}\big(\widetilde{\Psi}_\zeta^T\Psi_r^{*-T}\Lambda_\zeta^{-1}\dot{\widetilde{\Psi}}_\zeta + \widetilde{\Psi}_r^T\Psi_r^{*-T}\Lambda_r^{-1}\dot{\widetilde{\Psi}}_r\big) \end{aligned} \tag{B.5}$$

with this and, taking advantage of trace properties for square matrices $\mathbf{tr}(\mathbf{AB}) = \mathbf{tr}(\mathbf{BA})$, we can introduce the following adaptation laws:

$$\dot{\Psi}_\zeta = \dot{\widetilde{\Psi}}_\zeta = -\Lambda_\zeta \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{\zeta}^T - \mathbf{S} \Upsilon_{\boldsymbol{\zeta}} \bar{\Psi}_{\boldsymbol{\zeta}}$$
$$\dot{\Psi}_r = \dot{\widetilde{\Psi}}_r = -\Lambda_r \mathbf{B}_r^T \mathbf{P} \boldsymbol{e} \boldsymbol{r}^T - \mathbf{S} \Upsilon_{\mathrm{r}} \bar{\Psi}_{\mathrm{r}}$$

where Λ_ζ and Λ_r are positive definite matrices designed to tune convergence rate of the adaptive gains. Diagonal matrices $\mathbf{S}$, $\Psi_{\boldsymbol{\xi}}$, Ψ_{r} are activation and regulation matrices for Eq. 5.18 as explained in *Section 5.4*. $\mathbf{P}$ with $\mathbf{A}_r$ satisfy $\mathbf{PA}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$ as the necessary and sufficient stability condition for the reference model (Eq. 5.15). Putting these laws in Eq. B.5, the Lyapunov derivative simplifies to:

$$\dot{\mathbf{V}} = -\frac{1}{2} \boldsymbol{e}^T \mathbf{Q} \boldsymbol{e}$$

since $\mathbf{V} > 0$ and $\dot{\mathbf{V}} \leq 0$ the system in Eq. B.3 is stable from Lyapunov perspective, thus $\boldsymbol{e}$, $\widetilde{\Psi}_\zeta$, and $\widetilde{\Psi}_r$ are bounded. As $\boldsymbol{r}$ is bounded by definition, the system (Eq. B.3) is bounded as well. Therefore, $\ddot{\mathbf{V}} = -\boldsymbol{e}^T \mathbf{Q} \dot{\boldsymbol{e}}$ will be bounded always, and from Barbalat's lemma (Khalil and Grizzle (2002)), this system is asymptotically stable.

B.2 Contact-Frame Estimation

We train a model to estimate, online from tactile feedback, the frame of the contact, $\{C_i\} = \{\hat{\boldsymbol{n}}_i, \hat{\boldsymbol{t}}_i, \hat{\boldsymbol{o}}_i\}$. The tactile sensor consists of a rigid core surrounded by an elastic liquid-filled skin that provides compliance similar to the human fingertip. When forces are applied to the contact, the skin and fluid deforms, which is detected by an array of electrodes on the surface of the sensor core.

Model Training

We first need to collect the ground truth, consisting of (a) BioTac electrodes impedance, $\boldsymbol{\rho}_i \in \mathbb{R}^{19}$, (b) the orientation of fingertip base (sensor base), and (c) the orientation of the contact plane. We use the OptiTrack motion-capture system to get both fingers, $\boldsymbol{q}_{f,i}^O$, and contact plane, $\boldsymbol{q}_{c,i}^O$, quaternions: $\hat{\boldsymbol{n}}_{c,i}^B = \begin{bmatrix} -2(q'_{i,x}q'_{i,z} + q'_{i,w}q'_{i,y}) \\ -2(q'_{i,y}q'_{i,z} - q'_{i,w}q'_{i,x}) \\ 1 - 2(q'^2_{i,w} + q'^2_{i,z}) \end{bmatrix}$ in which $\boldsymbol{q}'_i = \boldsymbol{q}_{f,i}^O \bar{\boldsymbol{q}}_{c,i}^O = [q'_{i,w}, q'_{i,x}, q'_{i,y}, q'_{i,z}]$. The collected dataset is represented by $\mathscr{D} = \{\hat{\boldsymbol{n}}_{c,i}^{B\ (m)}, \boldsymbol{\rho}_i^{(m)}\}_{m=0}^{m=M}$ where M is the total number of samples(here 40,000); see an example of data collection and the recording setup in supplementary videos. The desired mapping, $\hat{\boldsymbol{n}}_{c,i}^B = f_{NN}(\boldsymbol{\rho}_i)$ is modeled and learned by an artificial neural network (ANN) (Bishop (1995)), and by using PyTorch library (Paszke et al. (2017)). The data is split into training (75%) and testing (25%) sets for 20 folds. The key hyper-parameters associated with the shallow neural networks are (a) activation

Table B.1: Results on training and testing the model for contact normal estimation. This model is selected after a 20-fold cross validation with training (75%) and testing (25%) sets.

Hidden layers	Neurons per layer	Training set absolute error	Test set absolute error
2	200	$1.81° \pm 1.40°$	$5.70° \pm 1.90°$

function, (b) number of hidden layers, and (c) neurons per each hidden layer. The activation function used in the networks is ReLU, and the other hyperparameters were determined using grid search. The model accuracy on their respective datasets are provided in Table B.1.

Using the ANN Model

The learned mapping $\hat{\boldsymbol{n}}^B_{c,i} = f_{NN}(\boldsymbol{\rho}_i)$ outputs the vector $\hat{\boldsymbol{n}}^B_{c,i}$ presented in the fingertip's base frame, which is needed to be transformed $\hat{\boldsymbol{n}}^B_{c,i}$ to the inertia frame $\{N\}$, fixed to the root frame of the robotic hand. We can use the forward kinematic of fingertips given the joint configuration $\boldsymbol{q}_i$ of the i-th fingertip :

$$\hat{\boldsymbol{n}}_i = \boldsymbol{R}_i^T(\boldsymbol{q}_i)\, f_{NN}(\boldsymbol{\rho}_i) \tag{B.6}$$

matrix $\boldsymbol{R}_i(\boldsymbol{q}_i)$ is the rotation matrix from frame $\{B\}_i$ to $\{N\}$. Finally, by using Eq. B.6, contact frame $\{C_i\} = \{\hat{\boldsymbol{n}}_i, \hat{\boldsymbol{t}}_i, \hat{\boldsymbol{o}}_i\}$ at i-th contact point are updated with the provided frequency of tactile sensors (60Hz for BioTac).

C Appendix of Chapter 6

C.1 EMG Motion Decoding

Commercially available RPHs rely on identifying motion intention using EMG. The control of prostheses is usually achieved by placing two electrodes on two remaining antagonist muscles of the forearm. A threshold is set on the EMG amplitude acquired at a fixed frequency to control one degree of freedom (DoF) at the time, closing or opening the fingers by a small increment (Mendez et al. (2021a)). This is insufficient to capture individuated finger motions and does not allow to provide continuous control of fingers. Finer EMG-based control can be obtained using single-finger angle regression (Liu et al. (2021)). With this method, the RPH follows the user's intended motion in an intuitive manner (Farina et al. (2014)). However, continuous control of finger motion with EMG is very sensitive to noise. The smallest error in detection of EMG intent could lead a finger to inadvertently re-open, letting the object slip. Controlling for a stable and robust grasp when manipulating objects is crucial to restore basic dexterity needed for everyday use. This requires ensuring fast and accurate control of finger-object interactions.

When integrating users' motor intentions for individuated finger control, a possible solution is to distribute the control by automating some parts of the motor commands and relieving the user from precise modulation (C.M. et al. (2002)). It can ease grasping through preshaping (Došen et al. (2010)), grip force adjustment (C.M. et al. (2002), slip detection (Tura A., Lamberti C, Davalli A (1998)), and even hand closing (Fani et al. (2016)). The Ottobock Sensorhand Speed is a commercial example of shared control in RPHs, which automatically increases thumb flexion during grasping in response to slippage (Ciancio et al. (2016)). A more recent study (Zhuang et al. (2019)) showed that an EMG-based shared control strategy could ensure safe handling of a bottle filled with content. This work leveraged an autonomous robot control that maximized the number of contacts between the robotic hand and an object.

C.1.1 EMG Setup and Model Calibration

In each session, the subjects have to calibrate the finger decoding model and the continuous shoulder controller. For the finger model, the subjects follow a series of single and multi-finger movements (alternated with a rest position) on a screen performed by a hand in a virtual environment developed in Unity. The sequence of movements is repeated 6 times, each movement is held for 5 s with a rest position between each movement of 3 s. The rationale is to dissociate three main states of the fingers: rest in a middle position without muscle activation, flexed and opened positions. The total calibration time for the finger movements is 9 minutes and 30 s. Hand kinematics is obtained from the finger angles of the virtual hand at 60 Hz and is synchronized with the EMG acquisition system. Six DoFs are recorded from the virtual hand that correspond to the flexion of each finger and the opposition of the thumb. Hand kinematics is scaled between 0 (open) and 10 (close), and the rest position is set to 3 for each DoF to mimic a resting hand pose.

To calibrate the continuous shoulder controller, the same setup is used, with the virtual hand alternating between closed (target = 10), opened (target = 0) and a middle position (target = 5) to record 1 DoF. The subjects are asked to elevate their shoulder when the hand is flexed, lower their shoulder when the hand is in the open position, and rest when the virtual hand is in the middle position. Each movement is held for 5 s, alternating with a rest position for 3 s. The sequence is repeated 6 times for a total calibration time of 1 minute and 35 s.

A sliding window of 200 ms is used with a moving step of 30 ms (170 ms overlap). To evaluate the offline performance of the decoders, the last repetition of the sequence of movements is used as a validation set. In total, 5 features are extracted to serve as input for the decoders (Boostani and Moradi (2003)): (i) mean absolute value, (ii) waveform length (cumulative length of the EMG waveform over time), (iii) maximum absolute value, (iv) zero crossings (number of times the signal crosses zero), and (v) slope sign change (number of times the slope of the signal changes sign).

For the finger decoder, a Multi-layer Perceptron (MLP) regressor is used to decode simultaneously the 6 DoFs. The features of the 6 forearm channels are used. This model already shows the possibility to decode single finger movements (Zhuang et al. (2019); Mendez et al. (2021b); Ngeo et al. (2014)). The model is designed by using Keras[1] with Tensorflow[2] backend, it has an architecture with 1 hidden layer with 32 nodes (ReLU activation function). It is trained with a drop rate of 0.2 using gradient descent with a batch size of 16 during 50 epochs. The learning rate is set to 0.01 and divided by 2 every 10 epochs. Early stopping is applied if the validation loss is not decreasing for more than 13 epochs. To take into account the delay between the movement of the virtual hand and the actual movement of the subject, the labels are shifted from 0 to 10 windows (delay of maximum 300 ms) through visual inspection in order to maximize validation performance. The shoulder continuous control is obtained by

[1] https://github.com/fchollet/keras
[2] https://www.tensorflow.org/

using an SVR algorithm with an RBF kernel from sklearn[1]. The features of the 2 shoulder EMG channels are used. This model is chosen instead of an MLP to avoid overfitting due to the lower amount of calibration data, since the decoding task was simpler. To maximize real-time performance, decoded values from the MLP are smoothed by using a 3-point median filter. Moreover, to reduce noise when the subjects are performing the tasks, predicted values are set to 0 if the decoded value is below 2 and set to 10 if the decoded value is higher than 7. Similar post-processing is applied on the SVR shoulder decoder with a median filter and value clipping, if the decoded values are out of bound.

C.2 Optimization-Based Robotic Hand Controller

C.2.1 Dynamic Hand Pose Adaptation

Here, we present our algorithms for dynamically online adapting the hand pose during the exploration process.

Modelling of the Hand

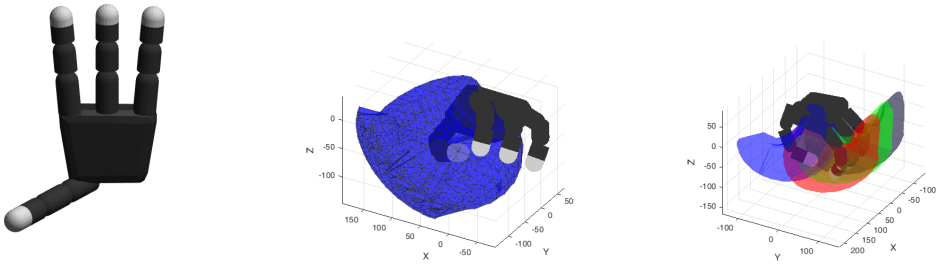
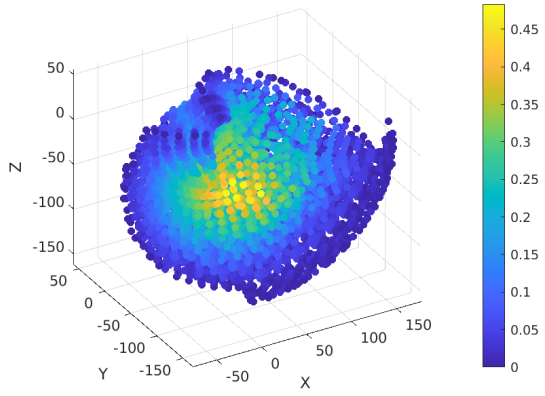


Figure C.1: Hand model and the sampled reachability map: (left) Allegro hand model, (middle) Sampled reachability map of thumb tip, and (right) Sampled reachability map of all fingertips.

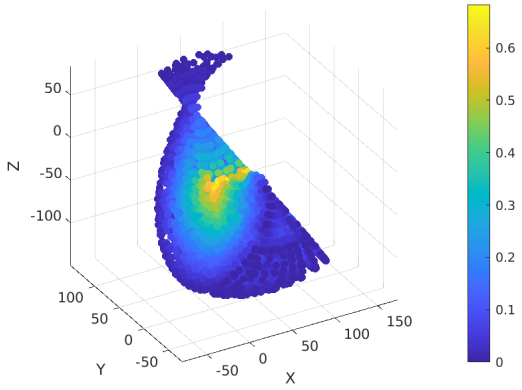
We use an Allegro hand (left) model to illustrate our algorithms. Allegro Hand (see Figure C.2) is a 16-DoF robotic hand that has four fingers, with each one having 4 DoFs: one ab- /adduction DOF and three extension/flexion DOFs. As a naming convention, we individually name the thumb, the index finger, the middle finger, and the ring finger as $F1$, $F2$, $F3$, and $F4$. Moreover, we use the superscription "tip" to indicate the phalanx of the fingertip. The geometric shape of finger phalanges is approximated using cylinders.

We represent all contacts on fingertips in the *hand reference frame*, denoted as **H**, which is located at the geometric center of the palm.

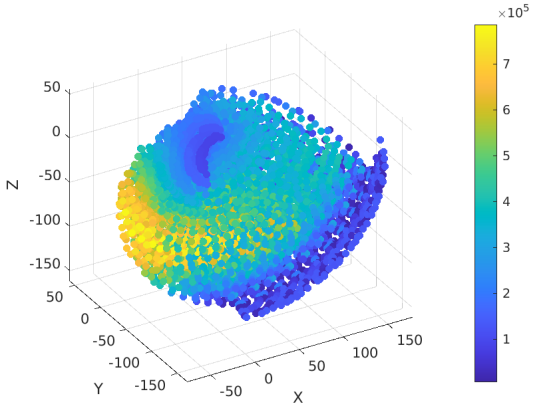
[1] https://scikit-learn.org/stable/



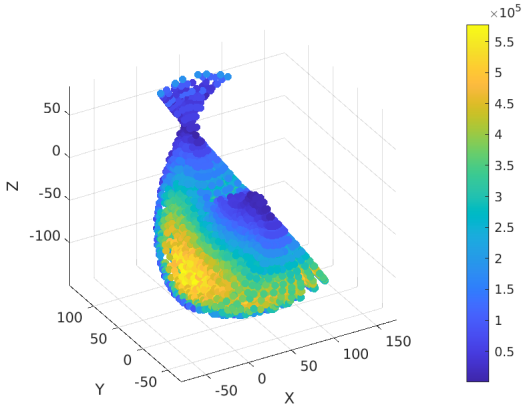
(a) Sampled isotropic reachability index map of the thumb



(b) Sampled isotropic reachability index map of the finger



(c) Sampled manipulability index map of the thumb



(d) Sampled manipulability index map of the finger

Figure C.2: Visualization of the sampled isotropic reachability index dataset and manipulability index dataset of thumb and finger, respectively. Each map consists of 10,000 sampled data points.

Modelling of contact

We enable a contact point $\mathbf{p}_i$ at an arbitrary position on the surface of the fingertip phalanx F_i^{tip}, $i = 1,\ldots,N_F$, with N_F being the number of fingers. For this purpose, we parameterize the contact using cylindrical coordinates:

- $\alpha_i \in [0,1]$, where $\alpha_i L$ is the distance from the contact point to the base of the fingertip phalanx of length L;
- $\rho_i \in \mathbb{R}^+$, the radical distance from the contact point to the central axis of the fingertip link;
- $\psi_i \in (-\pi,\pi]$, the angular coordinate.

Modelling of the object

We selected five everyday objects in different shapes and sizes for experimental validation: In the simulation, each object is represented by its 3D point cloud set.

Reachability map

The reachability map $\mathscr{R}_i$ is a set of all reachable Cartesian positions of the contact point $\mathbf{p}_i$ on the i^{th} finger digit. We sample uniformly in each joint's motion range and register each joint's corresponding Cartesian position to the joint's reachable surface. The three-dimensional spatial volume enclosed by the reachable surfaces of two adjacent joints is then the *reachability map* of the finger link. For example, Figure C.1 illustrates the reachable space of the thumb distal phalanx. Similarly, we can construct the reachability map set of the entire hand, $\{\mathscr{R}_i\}$, by iterating over all finger phalanges in hand (see Figure C.1). This map represents the maximum reachable space of the corresponding finger phalanx and can be used to generate the exploration target during exploration.

C.2.2 Finger motion planning

Given an estimation of the object model, the next exploration position can be determined based on the intersection region of the reachability map of the hand and the geometric shape of the object.

We use the vector $\mathbf{n}_i \in \mathscr{R}^3$ to indicate the surface normal direction at the contact point $\mathbf{p}_i$, and $\mathbf{t}_i \in \mathscr{R}^3$ the tangential direction of the surface at $\mathbf{p}_i$; both $\mathbf{n}_i$ and $\mathbf{t}_i$ are estimated by the Gaussian process model (see Sec. 6.2.1).

Then the next desired contact point of the current contact point $\mathbf{p}_i$ is calculated as:

$$\mathbf{p}_i^* = \mathbf{p}_i + d_s \cdot \mathbf{t}_i,\ i = 1,\ldots,N_F, \tag{C.1}$$

where d_s is the exploration step length, depending on the motion range of the robotic finger.

Problem formulation

We formulate the dynamic hand pose adaptation algorithm as a constrained optimization problem. This optimization problem aims at determining the desired exploration positions on the surface of the target object for each fingertip. The parameters of the hand are optimized in the framework such that the pose of the wrist is adjusted for each exploration step to improve the manipulability and the reachable space of fingers.

Objective function: The hand pose adaptation mainly serves two objectives: (i) increase the potential exploration region for each fingertip in the next step, and (ii) avoid moving any fingers to a singular position during exploration. We introduce two kinematic metrics in the corresponding aspects.

The isotropic reachability index: This index indicates whether a point inside the 3D spatial space of the reachability map has isotropic distances to the boundary of the reachability map in arbitrary directions:

$$\eta_i = \frac{d_{\min}(p_i, \mathscr{S}_i)}{d_{\max}(p_i, \mathscr{S}_i)},\ i = 1, 2, \ldots, N, \tag{C.2}$$

where p_i is the point of interest on the i^{th} finger digit, $d_{\min}(p_i, \mathscr{S}_i)$ and $d_{\max}(p_i, \mathscr{S}_i)$ are the minimum and maximum distance from p_i to the surface curve of the reachability map $\mathscr{S}_i$, respectively.

We train a Gaussian process regression model for each fingertip to enable fast online computation. We use the fingertip point $p_{i_{tip}}$ as the reference point p_i, to sample a set of η_i values given joint angle configurations. Then, we train the isotropic index model for each finger and thumb as

$$\eta_i = \mathscr{G}_\eta(\mathbf{q}_i),\ i = 1, 2, \ldots, N. \tag{C.3}$$

where $\mathbf{q}_i \in \mathbb{R}^4$ denotes the joint configuration of the i^{th} finger.

The manipulability index: We use the manipulability index (Yoshikawa (1985)) of each fingertip to indicate the quality of kinematic configuration of the contact point. It is defined as:

$$\omega_i = \sqrt{\det(JJ^\intercal)}, \tag{C.4}$$

where J is the Jacobian of the fingertip. A larger manipulability index value corresponds to a kinematic configuration further away from the singularity, enabling a more flexible movement.

Similar to the isotropic index model, we use GPR to train a manipulability index model for fast

online inference given a joint angle configuration $\mathbf{q}_i$:

$$\omega_i = \mathcal{G}_\omega(\mathbf{q}_i),\ i = 1, 2, \ldots, N. \tag{C.5}$$

Formulation of objective function: Finally, we formulate our objective function for dynamic hand pose adaptation, consisting of kinematic metrics from four aspects:

(i) $\Delta\mathbf{q}^T M_q \Delta\mathbf{q}$: damping of joint angles, to penalize excessive joint angle movements, where $\Delta\mathbf{q}$ indicates the change in joint angles with respect to the previous configuration;

(ii) $\delta^T M_\delta \delta$: damping of slack variables, to facilitate the satisfaction of soft constraints;

(iii) ω_i, $i = 1, \ldots, N$, the manipulability indices of all fingertips;

(iv) η_i, $i = 1, \ldots, N$, the isotropic reachability indices of all fingertips.

Therefore, the objective function is formulated as:

$$\mathcal{Q} = \Delta\mathbf{q}^T M_q \Delta\mathbf{q} + \delta^T M_\delta \delta + \sum_i^N \frac{1}{\omega_i} + \sum_i^N \frac{1}{\eta_i} \tag{C.6}$$

The objective function's value is minimized to guarantee a desired motion, and M_q and M_δ are weighting matrices.

Equality constraints: We use quaternion $\mathbf{q}^H = (q_x^H, q_y^H, q_z^H, q_w^H)$ to parameterize the orientation of the hand reference frame $\mathbb{H}$, and $\mathbf{q}^H$ subjects to:

$$\|\mathbf{q}^H\| = 1 \tag{C.7}$$

We introduce slack variables $\delta_i \in \mathbb{R}^3$ in the formulation of the equality constraints, to facilitate the contact point $\mathbf{p}_i \in \mathbb{R}^3$ to be as close as possible to the desired contact point $\mathbf{p}_i^* \in \mathbb{R}^3$ for each finger i. We formulate this as a soft constraint, providing freedom for the planning algorithm to compensate for model uncertainty.

$$\mathbf{p}_i - \mathbf{p}_i^* + \delta_i = 0,\ i = 1, 2, \ldots, N. \tag{C.8}$$

Inequality constraints: The problem subjects to multiple nonlinear inequality constraints, listed as follows.

We use $\mathbf{Q}_i$ to indicate the parameter set that contains all variables parameterizing the contact point $\mathbf{p}_i$:

$$\mathbf{Q}_i = (\mathbf{q}^H, \mathbf{q}_i, \alpha_i, \psi_i, \phi_i) \tag{C.9}$$

$$q_i \in [\underline{q}, \overline{q}],\ \forall q \in \mathbf{Q}_i,\ i = 1, \ldots, N, \tag{C.10}$$

with q being a generalized parameter in $\mathbf{Q}_i$, $\underline{q}$ and $\overline{q}$ being the corresponding lower and upper bound of q.

The hand palm surface must face the opposite direction as the surface normal of the local region during exploration. Therefore:

$$\frac{\langle \mathbf{n}^H, \mathbf{n}^O \rangle}{\|\mathbf{n}^H\| \|\mathbf{n}^O\|} < 0 \tag{C.11}$$

where $\mathbf{n}^H$ is the normal direction of hand palm, and $\mathbf{n}^O$ is the local object surface normal, estimated based on current contact points.

We formulate nonlinear inequality constraints for all fingertips to force the hand into a collision-free configuration. Since all fingertips are in contact and move on the object surface during exploration, the distance between any two fingertips F_i^{tip} and F_j^{tip} must satisfy:

$$d(F_i^{tip}, F_j^{tip}) > r_i + r_j, i = 1, 2, \ldots, N, j = 1, 2, \ldots, N, i \neq j, \tag{C.12}$$

where r is the radius of the finger phalanx (approximated as cylinder). We use a cylinder to approximate the geometry of a fingertip digit. Hence, we calculate the distance between two finger digits as a sequence of N pairwise distances between two uniformly-sampled points on the central axis of each cylinder. Therefore, $d(F_i^{tip}, F_j^{tip}) \in \mathbb{R}^N$, and $d_n = d(c_i, c_j) > r_i + r_j$, $n = 1, \ldots, N$, where c_i and c_j represent sampled points on the central axis of F_i^{tip} and F_j^{tip}, respectively.

Optimization variables: The optimization variables Θ consist of joint angles of both the wrist and the fingers.

- Joint angles $\mathbf{q}_i$ and cylindrical coordinates $(\alpha_i, \rho_i, \psi_i)$ that parameterize all contact points $\mathbf{p}_i$, $i = 1, 2, \ldots, N$;
- Hand pose position $\mathbf{p}^H \in \mathbb{R}^3$ and orientation $\mathbf{q}^H \in \mathbb{R}^4$ (quaternion);
- Slack variables used in soft constraints, $\delta_i \in \mathbb{R}^3$ is the slack variable associated with $\mathbf{p}_i$.

Problem formulation: To summarize, we formulate the hand pose adaptation problem as follows:

$$\begin{aligned} &\Theta^* = \arg\min_{\Theta}\ \mathscr{Q} \\ &\text{subject to } (C.7),\ (C.8),\ (C.10),\ (C.11),\ (C.12) \end{aligned} \tag{C.13}$$

Figure C.3: Exploration of the experimental object bottle by following four different experimental protocols. Each row represents the exploration following one protocol. From top to bottom: Exp.I, II, III, and IV. Figures ranged in each column are taken at the same time step: 0 (initial), 100, 200, 300, and 400. The trajectories of F_1^{tip}, F_2^{tip}, F_3^{tip}, and F_4^{tip} are illustrated in red, green, blue, and yellow colors, respectively.

C.2.3 Validation of Dynamic Hand Pose Adaptation

In this section, we explain the experimental steps.

Initialization

At the beginning of the exploration process, the target object is being placed on a table surface, in front of the robotic system. We consider the region on the table surface as the *region of interest* (ROI) for the exploration task. The robotic hand then opens up fingers, and moves toward the center of the workspace.

Table C.1: Experiments for evaluation of the proposed dynamic hand pose adaptation algorithm.

Experiment Type	Hand Pose Regulation	Finger Kinematics Optimization
I	✗	✗
II	✗	✓
III	✓	✗
IV	✓	✓

Establishing contact

The robotic hand opens fingers, moves toward the ROI, until all fingers detected contacts. These contact points are used for updating the GPR model.

In the simulation experiment, we simply selected points from the object point cloud, which are closest to the initial spatial position of each fingertip, respectively. Then, our online hand pose adaptation algorithm solves for a feasible configuration for the hand pose to achieve the initial contacts. This guarantees the same initial condition for all simulation scenarios, hence easier for us to compare the performance in subsequent experimental steps.

Online exploration

At every step, the current detected contact points are used to update the GPR model; the GPR model then generates the surface normal vector $\mathbf{n}_i$ and $\mathbf{t}_i$ at the contact point of each fingertip, which are then used to generate the desired contact points for the next exploration step (see Eq. C.1). In simulation, we take advantage of the object point cloud to assist the evaluation of our algorithm. If the calculated next contact points belong to the object point cloud, we simply use these points as targets. Otherwise, we select the nearest neighbor of the desired contact points from the object point cloud. If the desired target point can be successfully achieved by solving the optimization problem, we label the point as explored. Otherwise, the actual position of the fingertip after the exploration step is registered in the object point cloud.

The robotic hand iterates this exploration step, until the object has been extensively explored, or the uncertainty of the constructed GP model has dropped below a threshold.

Procedure

Our proposed dynamic hand pose adaptation algorithm consists of two main aspects: (1) the regulation of hand pose, and (2) the improvement of finger kinematics. Therefore, we design four types of experiments to validate the effectiveness of our proposed algorithm, summarized in Table C.1.

In the case where the regulation of hand pose is absent, we consider the hand pose can move to different spatial positions, but its orientation remains constant as the initial state. For this purpose, we simply exclude the quaternion used for describing hand poses from the optimization variable set.

When the optimization of finger kinematics is disabled, we remove the terms of manipulability ($\sum_i^N \omega_i$), and isotropic reachability index ($\sum_i^N \eta_i$) from the objective function (Eq. C.6).

C.2.4 Results

We conducted experiments by exploring a bottle, following each of these four experimental types. Figure C.3 shows the experimental results.

In the first row (see Figure C.3(I-1) to (I-5)), the hand explored the object surface with neither hand pose regulation nor finger kinematics optimization. The hand collided with the object model early within 100 steps (see Figure C.3(I-2)); and this collision exist in almost all steps in the remaining exploration. Moreover, in some steps, the fingers failed to establish contacts on the object surface, indicated by the scattered points outside the object surface model (see Figure C.3(I-5)).

The figures in the second row (see Figure C.3(II-1) to (II-5)) correspond to the results from experiment type II. In this experiment, the hand pose is not regulated, but the kinematics of fingers are optimized in each experimental step. In comparison to experiment type I, it is rare that the fingers will collide with the explored object model. Nevertheless, the hand palm still collides with the experimental object model (see Figure C.3(II-3) and (II-4)).

In experiment type III (see Figure C.3(III-1) to (III-5)), the hand pose was constantly regulated in each experimental step. Therefore, the hand does not collide with the object point cloud anymore. Instead, the orientation of the palm keeps changing to adapt to the local surface curvature of the explored object. As the optimization of finger kinematics is still absent, in some exploration steps (see Figure C.3(III-2) and (III-4)), the fingers either collide with the object model, or close to singular configuration (see the thumb in Figure C.3(III-2)).

Hand in experiment type IV was controlled by our proposed dynamic hand pose adaptation algorithm. The robotic hand was able to establish stable, collision-free contacts on the surface of the object in most steps. Moreover, all fingers contacted the object in natural configurations that are far from any joint singularities.

Finally, we applied our proposed algorithm to explore different experimental objects. Figure C.4 illustrate the exploration process.

In most cases, the hand was able to adapt its orientation to the object local curvature, with all fingertips in contact with the object surface. Collision between hand and object still exists in certain cases, mainly due to the abrupt change of the object's local curvature. For example, at

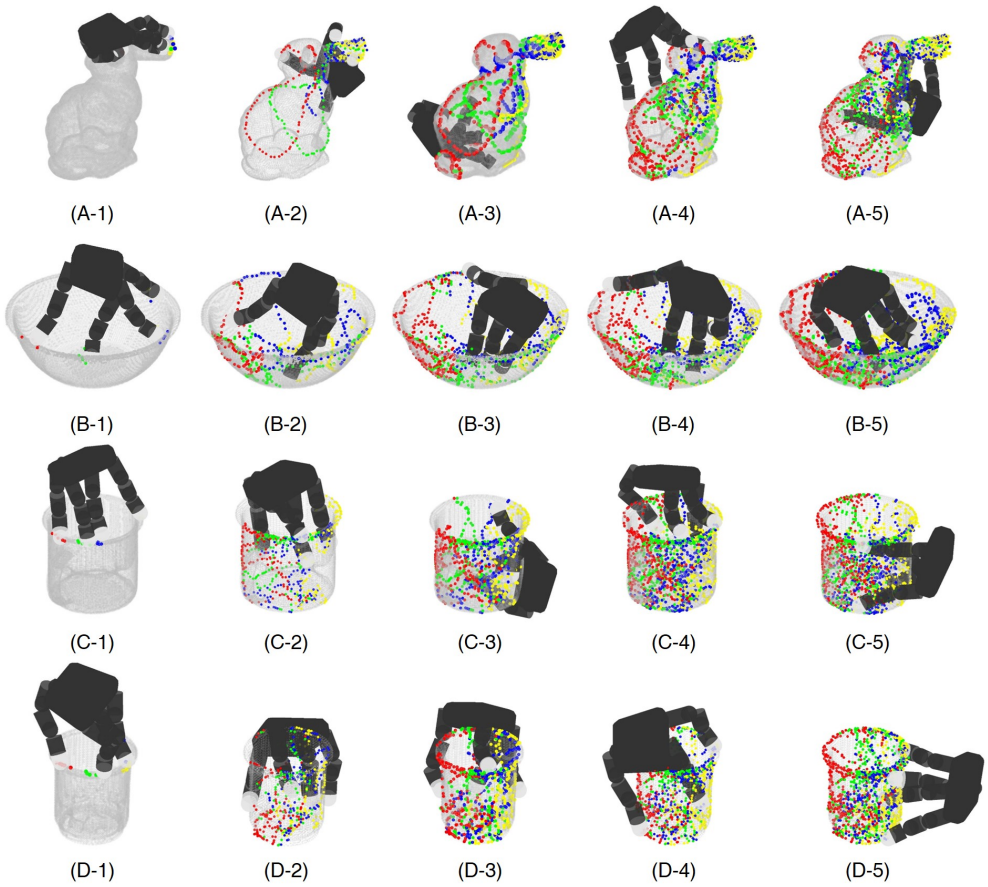


Figure C.4: Simulated exploration of all experimental objects. From top to bottom: bunny, bowl, cup, mug. Figures in each row show the hand configuration of exploring one object, taken at different time steps. From left to right: initial state, steps of 100, 200, 300, and 400. The trajectories of F_1^{tip}, F_2^{tip}, F_3^{tip}, and F_4^{tip} are illustrated in red, green, blue, and yellow colors, respectively.

the edge of the cup, or close to the handle of the mug.

Bibliography

Abe, Y., Da Silva, M., and Popović, J. (2007). Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 249–258.

Ajallooeian, M., van den Kieboom, J., Mukovskiy, A., Giese, M. A., and Ijspeert, A. J. (2013). A general family of morphed nonlinear phase oscillators with arbitrary limit cycle shape. *Physica D: Nonlinear Phenomena*, 263:41–56.

Anderson, B. D. and Moore, J. B. (2007). *Optimal control: linear quadratic methods*. Courier Corporation.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Azimi, V. et al. (2019). Model-based adaptive control of transfemoral prostheses: Theory, simulation, and experiments. *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, 51(2):1174–1191.

Baerlocher, P. and Boulic, R. (2004). An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The visual computer*, 20(6):402–417.

Bai, Y. and Liu, C. K. (2014). Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565. IEEE.

Belongie, S. et al. (1999). Rodrigues' rotation formula. *From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld. wolfram. com/RodriguesRotation-Formula. html*.

Bensmaia, S. J., Tyler, D. J., and Micera, S. (2020). Restoration of sensory information via bionic hands.

Berenson, D., Srinivasa, S., and Kuffner, J. (2011). Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460.

Bicchi, A. (2000). Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Transactions on robotics and automation*, 16(6):652–662.

Bierlaire, M. (2015). *Optimization: Principles and Algorithms*. EPFL Press, Lausanne.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA.

Björkman, M., Bekiroglu, Y., Högman, V., and Kragic, D. (2013). Enhancing visual perception of shape through tactile glances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3180–3186. IEEE.

Boostani, R. and Moradi, M. H. (2003). Evaluation of the forearm EMG signal features for the control of a prosthetic hand. *Physiological Measurement*, 24(2):309–319.

Bouyarmane, K. and Kheddar, A. (2011). Multi-contact stances planning for multiple agents. In *2011 IEEE International Conference on Robotics and Automation*, pages 5246–5253. IEEE.

Bouyarmane, K. and Kheddar, A. (2017). On weight-prioritized multitask control of humanoid robots. *IEEE Trans. on Automatic Control*, 63(6):1632–1647.

Buchli, J., Kalakrishnan, M., Mistry, M., Pastor, P., and Schaal, S. (2009). Compliant quadruped locomotion over rough terrain. In *2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 814–820. IEEE.

Burdet, E. and Codourey, A. (1998a). Evaluation of parametric and nonparametric nonlinear adaptive controllers. *Robotica*, 16(1):59–73.

Burdet, E. and Codourey, A. (1998b). Evaluation of parametric and nonparametric nonlinear adaptive controllers. *Robotica*, 16(1):59–73.

Caccavale, F. and Uchiyama, M. (2016). Cooperative manipulation. In *Springer Handbook of Robotics*, pages 989–1006. Springer.

Calandra, R., Ivaldi, S., Deisenroth, M. P., Rueckert, E., and Peters, J. (2015). Learning inverse dynamics models with contacts. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3186–3191. IEEE.

Camoriano, R., Traversaro, S., Rosasco, L., Metta, G., and Nori, F. (2016). Incremental semi-parametric inverse dynamics learning. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 544–550.

Castiello, U. (2005). The neuroscience of grasping. *Nature Reviews Neuroscience 2005 6:9*, 6(9):726–736.

Chatzilygeroudis, K. and Mouret, J.-B. (2018). Using parameterized black-box priors to scale up model-based policy search for robotics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE.

Chatzilygeroudis, K., Rama, R., Kaushik, R., Goepp, D., Vassiliades, V., and Mouret, J.-B. (2017). Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58. IEEE.

Chatzilygeroudis, K., Vassiliades, V., Stulp, F., Calinon, S., and Mouret, J.-B. (2019). A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*.

Chatzilygeroudis, K., Vassiliades, V., Stulp, F., Calinon, S., and Mouret, J.-B. (2020). A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*, 36(2):328–347.

Chavan-Dafle, N. and Rodriguez, A. (2015). Prehensile pushing: In-hand manipulation with push-primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6215–6222. IEEE.

Chavan-Dafle, N. and Rodriguez, A. (2018). Stable prehensile pushing: In-hand manipulation with alternating sticking contacts. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 254–261. IEEE.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. (2019). Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE.

Chen, C., Liu, Z., Zhang, Y., and Xie, S. (2017). Coordinated motion/force control of multi-arm robot with unknown sensor nonlinearity and manipulated object's uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1123–1134.

Chowdhary, G., Kingravi, H. A., How, J. P., and Vela, P. A. (2014). Bayesian nonparametric adaptive control using gaussian processes. *IEEE transactions on neural networks and learning systems*, 26(3):537–550.

Chung, S.-J. and Slotine, J.-J. E. (2009). Cooperative robot control and concurrent synchronization of lagrangian systems. *IEEE transactions on Robotics*, 25(3):686–700.

Ciancio, A. L., Cordella, F., Barone, R., Romeo, R. A., Bellingegni, A. D., Sacchetti, R., Davalli, A., Di Pino, G., Ranieri, F., Di Lazzaro, V., et al. (2016). Control of prosthetic hands via the peripheral nervous system. *Frontiers in neuroscience*, 10:116.

Cler, M. J., Michener, C. M., and Stepp, C. E. (2014). Discrete vs. continuous surface electromyographic interface control. *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014*, pages 4374–4377.

C.M., L., P.H., C., B., H., and K., E. (2002). Intelligent multifunction myoelectric control of hand prostheses. *Journal of Medical Engineering and Technology*, 26(4):139–146.

Collette, C., Micaelli, A., Andriot, C., and Lemerle, P. (2007). Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 81–88. IEEE.

Coros, S., Beaudoin, P., and Van de Panne, M. (2010). Generalized biped walking control. *ACM Transactions on Graphics (TOG)*, 29(4):1–9.

Craig, J. J. (1989). *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition.

Culbertson, P. and Schwager, M. (2018). Decentralized adaptive control for collaborative manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 278–285. IEEE.

Culbertson, P., Slotine, J.-J., and Schwager, M. (2021). Decentralized adaptive control for collaborative manipulation of rigid bodies. *IEEE Trans. on Robotics*, 37(6):1906–1920.

Cully, A., Chatzilygeroudis, K., Allocati, F., and Mouret, J.-B. (2018). Limbo: A flexible high-performance library for gaussian processes modeling and data-efficient optimization. *Journal of Open Source Software*, 3(26).

Dafle, N. C., Rodriguez, A., Paolini, R., Tang, B., Srinivasa, S. S., Erdmann, M., Mason, M. T., Lundberg, I., Staab, H., and Fuhlbrigge, T. (2014). Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585. IEEE.

D'Anna, E., Valle, G., Mazzoni, A., Strauss, I., Iberite, F., Patton, J., Petrini, F. M., Raspopovic, S., Granata, G., Iorio, R. D., Controzzi, M., Cipriani, C., Stieglitz, T., Rossini, P. M., and Micera, S. (2019). A closed-loop hand prosthesis with simultaneous intraneural tactile and position feedback. *Science Robotics*, 4(27).

De Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. *ACM Transactions on Graphics (TOG)*, 29(4):1–10.

Deisenroth, M. P., Fox, D., and Rasmussen, C. E. (2013). Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423.

Dollar, A. M. and Howe, R. D. (2007). Simple, robust autonomous grasping in unstructured environments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4693–4700. IEEE.

Doosti, B., Naha, S., Mirbagheri, M., and Crandall, D. J. (2020). Hope-net: A graph-based model for hand-object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6608–6617.

Došen, S., Cipriani, C., Kostić, M., Controzzi, M., Carrozza, M. C., and Popovič, D. B. (2010). Cognitive vision system for control of dexterous prosthetic hands: Experimental evaluation. *Journal of NeuroEngineering and Rehabilitation*, 7(1):1–14.

Dragiev, S., Toussaint, M., and Gienger, M. (2011). Gaussian process implicit surfaces for shape estimation and grasping. In *2011 IEEE International Conference on Robotics and Automation*, pages 2845–2850. IEEE.

Driess, D., Englert, P., and Toussaint, M. (2017). Active learning with query paths for tactile object shape exploration. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 65–72. IEEE.

Driess, D., Hennes, D., and Toussaint, M. (2019). Active multi-contact continuous tactile exploration with gaussian process differential entropy. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7844–7850. IEEE.

Du, Z., Iravani, P., and Sahinkaya, M. N. (2014). A new approach to design optimal excitation trajectories for parameter estimation of robot dynamics. In *2014 UKACC International Conference on Control (CONTROL)*, pages 389–394.

Ernesti, J., Righetti, L., Do, M., Asfour, T., and Schaal, S. (2012). Encoding of periodic and their transient motions by a single dynamic movement primitive. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 57–64. IEEE.

Escande, A., Mansard, N., and Wieber, P.-B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028.

Fani, S., Bianchi, M., Jain, S., Neto, J. S. P., Boege, S., Grioli, G., Bicchi, A., and Santello, M. (2016). Assessment of myoelectric controller performance and kinematic behavior of a novel soft synergy-inspired robotic hand for prosthetic applications. *Frontiers in Neurorobotics*, 10(October):1–15.

Farina, D., Jiang, N., Rehbaum, H., Holobar, A., Graimann, B., Dietl, H., and Aszmann, O. C. (2014). The extraction of neural information from the surface emg for the control of upper-limb prostheses: emerging avenues and challenges. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(4):797–809.

Feix, T., Romero, J., Schmiedmayer, H.-B., Dollar, A. M., and Kragic, D. (2016). The grasp taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77.

Feng, S., Whitman, E., Xinjilefu, X., and Atkeson, C. G. (2014). Optimization based full body control for the atlas robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 120–127. IEEE.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. (2008). An hdp-hmm for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pages 312–319. ACM.

Furukawa, N., Namiki, A., Taku, S., and Ishikawa, M. (2006). Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 181–187. IEEE.

Gams, A., Ude, A., and Morimoto, J. (2015). Accelerating synchronization of movement primitives: Dual-arm discrete-periodic motion of a humanoid robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2754–2760. IEEE.

Gandler, G. Z., Ek, C. H., Björkman, M., Stolkin, R., and Bekiroglu, Y. (2020). Object shape estimation and modeling, based on sparse gaussian process implicit surfaces, combining visual data and tactile exploration. *Robotics and Autonomous Systems*, 126:103433.

Gautier, M. and Khalil, W. (1991). Exciting trajectories for the identification of base inertial parameters of robots. In *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pages 494–499 vol.1.

Gerardo-Castro, M. P., Peynot, T., and Ramos, F. (2015). Laser-radar data fusion with gaussian process implicit surfaces. In *Field and Service Robotics*, pages 289–302. Springer.

Gijsberts, A. and Metta, G. (2012). Real-time model learning using incremental sparse spectrum gaussian process regression. *Neural Networks*.

Goodwin, G. C. and Sin, K. S. (2014). *Adaptive filtering prediction and control*. Courier Corporation.

Gribovskaya, E., Khansari Zadeh, S. M., and Billard, A. (2011). Learning non-linear multivariate dynamics of motion in robotic manipulators. *International Journal of Robotics Research*, 30:80–117.

Han, L. and Trinkle, J. C. (1998). Dextrous manipulation by rolling and finger gaiting. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 730–735. IEEE.

Hang, K., Bircher, W. G., Morgan, A. S., and Dollar, A. M. (2020). Hand–object configuration estimation using particle filters for dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(14):1760–1774.

Hershey, J. R. and Olsen, P. A. (2007). Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–317. IEEE.

Higham, N. J. (1990). Analysis of the cholesky decomposition of a semi-definite matrix.

Hitzler, K., Meier, F., Schaal, S., and Asfour, T. (2019). Learning and adaptation of inverse dynamics models: A comparison. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 491–498. IEEE.

Hogan, N. and Buerger, S. P. (2018). Impedance and interaction control. In *Robotics and automation handbook*, pages 375–398. CRC press.

Hollerbach, J., Khalil, W., and Gautier, M. (2016). Model identification. In *Springer handbook of robotics*, pages 113–138. Springer.

Ioannou, P. A. and Sun, J. (2012). *Robust adaptive control*. Courier Corporation.

Kaboli, M., Feng, D., Yao, K., Lanillos, P., and Cheng, G. (2017). A tactile-based framework for active object learning and discrimination using multimodal robotic skin. *IEEE Robotics and Automation Letters*, 2(4):2143–2150.

Karayiannidis, Y., Smith, C., Kragic, D., et al. (2016). Adaptive control for pivoting with visual and tactile feedback. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 399–406. IEEE.

Kastritsi, T., Dimeas, F., and Doulgeri, Z. (2018). Progressive automation with dmp synchronization and variable stiffness control. *IEEE Robotics and Automation Letters*, 3(4):3789–3796.

Khadivar, F., Gupta, S., Amanhoud, W., and Billard, A. (2021a). Efficient configuration exploration in inverse dynamics acquisition of robotic manipulators. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1934–1941. IEEE.

Khadivar, F., Lauzana, I., and Billard, A. (2021b). Learning dynamical systems with bifurcations. *Robotics and Autonomous Systems*, 136:103700.

Khalil, H. K. and Grizzle, J. W. (2002). *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ.

Khansari-Zadeh, S. M. and Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957.

Ko, J. and Fox, D. (2008). Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. In *IROS*.

Kocijan, J., Murray-Smith, R., Rasmussen, C. E., and Girard, A. (2004). Gaussian process model based predictive control. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2214–2219 vol.3.

Kontoudis, G. P., Liarokapis, M., and Vamvoudakis, K. G. (2019). An Adaptive, Humanlike Robot Hand with Selective Interdigitation: Towards Robust Grasping and Dexterous, In-Hand Manipulation. *IEEE-RAS International Conference on Humanoid Robots*, 2019-Octob:251–258.

Kronander, K. and Billard, A. (2015). Passive interaction control with dynamical systems. *IEEE Robotics and Automation Letters*, 1(1):106–113.

Kumar, V., Gupta, A., Todorov, E., and Levine, S. (2016). Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095*.

KyungYou, K.-J., Rhee, K.-W., and Shin, H.-C. (2010). Finger Motion Decoding Using EMG Signals Corresponding Various Arm Postures. *Experimental Neurobiology*, 19(1):54.

Lee, B., Zhang, C., Huang, Z., and Lee, D. D. (2019). Online continuous mapping using gaussian process implicit surfaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6884–6890. IEEE.

Lee, J., Grey, M. X., Ha, S., Kunz, T., Jain, S., Ye, Y., Srinivasa, S. S., Stilman, M., and Liu, C. K. (2018). DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500.

Lee, S. H., Suh, I. H., Calinon, S., and Johansson, R. (2015). Autonomous framework for segmenting robot trajectories of manipulation task. *Autonomous robots*, 38(2):107–141.

Li, M., Bekiroglu, Y., Kragic, D., and Billard, A. (2014). Learning of grasp adaptation through experience and tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3339–3346. Ieee.

Li, M., Hang, K., Kragic, D., and Billard, A. (2016a). Dexterous grasping under shape uncertainty. *Robotics and Autonomous Systems*, 75:352–364.

Li, Z., Deng, J., Lu, R., Xu, Y., Bai, J., and Su, C.-Y. (2016b). Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):740–749.

Li, Z., Ge, Q., Ye, W., and Yuan, P. (2016c). Dynamic balance optimization and control of quadruped robot systems with flexible joints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(10):1338–1351.

Limón, D. et al. (2006). On the stability of constrained mpc without terminal constraint. *IEEE Trans. on automatic control*, 51(5):832–836.

Liu, Y., Zhang, S., and Gowda, M. (2021). NeuroPose: 3D hand pose tracking using EMG wearables. *The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021*, pages 1471–1482.

Lober, R., Padois, V., and Sigaud, O. (2016). Efficient reinforcement learning for humanoid whole-body control. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 684–689. IEEE.

Meier, F., Hennig, P., and Schaal, S. (2014). Incremental local gaussian regression. In *Advances in Neural Information Processing Systems*, pages 972–980.

Meier, F., Kappler, D., Ratliff, N., and Schaal, S. (2016). Towards robust online inverse dynamics learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4034–4039. IEEE.

Meier, F., Kappler, D., Ratliff, N., and Schaal, S. (2016). Towards robust online inverse dynamics learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4034–4039.

Melchiorri, C. and Kaneko, M. (2016). Robot hands. In *Springer Handbook of Robotics*, pages 463–480. Springer.

Mendez, V., Iberite, F., Shokur, S., and Micera, S. (2021a). Current solutions and future trends for robotic prosthetic hands. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:595–627.

Mendez, V., Pollina, L., Artoni, F., and Micera, S. (2021b). Deep learning with convolutional neural network for proportional control of finger movements from surface EMG recordings. *International IEEE/EMBS Conference on Neural Engineering, NER*, 2021-May:1074–1078.

Mirrazavi Salehian, S. S., Figueroa, N., and Billard, A. (2018). A unified framework for coordinated multi-arm motion planning. *The International Journal of Robotics Research*, 37(10):1205–1232.

Modugno, V., Neumann, G., Rueckert, E., Oriolo, G., Peters, J., and Ivaldi, S. (2016). Learning soft task priorities for control of redundant robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 221–226. IEEE.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nakanishi, J., Cory, R., Mistry, M., Peters, J., and Schaal, S. (2008). Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757.

Neumann, K. and Steil, J. J. (2015). Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, 70:1–15.

Ngeo, J. G., Tamei, T., and Shibata, T. (2014). Continuous and simultaneous estimation of finger kinematics using inputs from an EMG-to-muscle activation model. *Journal of NeuroEngineering and Rehabilitation*, 11(1):1–14.

Nguyen-Tuong, D. and Peters, J. (2010). Using model knowledge for learning inverse dynamics. In *2010 IEEE international conference on robotics and automation*, pages 2677–2682. IEEE.

Nguyen-Tuong, D. and Peters, J. (2010). Using model knowledge for learning inverse dynamics. In *2010 IEEE International Conference on Robotics and Automation*, pages 2677–2682.

Nguyen-Tuong, D. and Peters, J. (2011). Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340.

Nguyen-Tuong, D., Peters, J., Seeger, M., and Schölkopf, B. (2008a). Learning inverse dynamics: A comparison. In *Advances in Computational Intelligence and Learning: Proceedings of the European Symposium on Artificial Neural Networks*, pages 13–18, Evere, Belgium. Max-Planck-Gesellschaft, d-side.

Nguyen-Tuong, D., Peters, J. R., and Seeger, M. (2009). Local gaussian process regression for real time online model learning. In *Advances in neural information processing systems*, pages 1193–1200.

Nguyen-Tuong, D., Seeger, M., and Peters, J. (2008b). Computed torque control with nonparametric regression models. *2008 American Control Conference*, pages 212–217.

Nguyen-Tuong, D., Seeger, M., and Peters, J. (2008c). Computed torque control with nonparametric regression models. In *2008 American Control Conference*, pages 212–217. IEEE.

Niekum, S., Osentoski, S., Konidaris, G., and Barto, A. G. (2012). Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246. IEEE.

Ogata, K. and Yang, Y. (2002). *Modern control engineering*, volume 4. Prentice hall India.

Okamura, A. M., Smaby, N., and Cutkosky, M. R. (2000). An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE.

Ortega, R. and Spong, M. W. (1989). Adaptive motion control of rigid robots: A tutorial. *Automatica*, 25(6):877–888.

Ottenhaus, S., Renninghoff, D., Grimm, R., Ferreira, F., and Asfour, T. (2019). Visuo-haptic grasping of unknown objects based on gaussian process implicit surfaces and deep learning. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 402–409. IEEE.

Ozawa, R. and Tahara, K. (2017). Grasp and dexterous manipulation of multi-fingered robotic hands: a review from a control view point. *Advanced Robotics*, 31(19-20):1030–1050.

Park, K. H., Suk, H. I., and Lee, S. W. (2016). Position-independent decoding of movement intention for proportional myoelectric interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(9):928–939.

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

Perrin, N. and Schlehuber-Caissier, P. (2016). Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59.

Pfanne, M., Chalon, M., Stulp, F., Ritter, H., and Albu-Schäffer, A. (2020). Object-level impedance control for dexterous in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(2):2987–2994.

Prattichizzo, D. and Trinkle, J. C. (2016). Grasping. In *Springer handbook of robotics*, pages 955–988. Springer.

Presse, C. and Gautier, M. (1993). New criteria of exciting trajectories for robot identification. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 907–912 vol.3.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. (2017). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

Rana, M. A., Li, A., Fox, D., Boots, B., Ramos, F., and Ratliff, N. (2020). Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. *arXiv preprint arXiv:2005.13143*.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge.

Righetti, L. and Schaal, S. (2012). Quadratic programming for inverse dynamics with optimal distribution of contact forces. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 538–543. IEEE.

Roa, M. A. and Suárez, R. (2015). Grasp quality measures: review and performance. *Autonomous robots*, 38(1):65–88.

Rognini, G., Sengül, A., Aspell, J., Salomon, R., Bleuler, H., and Blanke, O. (2013). Visuotactile integration and body ownership during self-generated action. *European Journal of Neuroscience*, 37(7):1120–1129.

Salini, J., Padois, V., and Bidaud, P. (2011). Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions. In *2011 IEEE International Conference on Robotics and Automation*, pages 1283–1290. IEEE.

Schiilkop, P., Burgest, C., and Vapnik, V. (1995). Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery & Data Mining. AAAI Press, Menlo Park, CA*, pages 252–257.

Schölkopf, B., Simard, P., Smola, A. J., and Vapnik, V. (1998). Prior knowledge in support vector kernels. In *Advances in neural information processing systems*, pages 640–646.

Seshagiri, S. and Khalil, H. K. (2000). Output feedback control of nonlinear systems using rbf neural networks. *IEEE Transactions on Neural Networks*, 11(1):69–79.

Sharifi, M. et al. (2021). Adaptive cpg-based gait planning with learning-based torque estimation and control for exoskeletons. *IEEE Robotics and Automation Letters*, 6(4):8261–8268.

Shi, J., Woodruff, J. Z., Umbanhowar, P. B., and Lynch, K. M. (2017). Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795.

Sodhi, P., Kaess, M., Mukadam, M., and Anderson, S. (2021). Learning tactile models for factor graph-based estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13686–13692. IEEE.

Sommer, N. and Billard, A. (2016). Multi-contact haptic exploration and grasping with tactile sensors. *Robotics and autonomous systems*, 85:48–61.

Sommer, N., Li, M., and Billard, A. (2014). Bimanual compliant tactile exploration for grasping unknown objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6400–6407. IEEE.

Spitz, J., Bouyarmane, K., Ivaldi, S., and Mouret, J.-B. (2017). Trial-and-error learning of repulsors for humanoid qp-based whole-body control. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 468–475. IEEE.

Spong, M. W. and Vidyasagar, M. (2008). *Robot dynamics and control*. John Wiley & Sons.

Stephens-Fripp, B., Alici, G., and Mutlu, R. (2018). A review of non-invasive sensory feedback methods for transradial prosthetic hands. *IEEE Access*, 6:6878–6899.

Stürz, Y. R., Affolter, L. M., and Smith, R. S. (2017). Parameter identification of the kuka lbr iiwa robot including constraints on physical feasibility. *IFAC-PapersOnLine*, 50:6863–6868.

Sun, C., He, W., Ge, W., and Chang, C. (2017). Adaptive neural network control of biped robots. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(2):315–326.

Sundaralingam, B. and Hermans, T. (2019). Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation. *Autonomous Robots*, 43(2):469–483.

Swevers, J., Ganseman, C., Tukel, D. B., de Schutter, J., and Van Brussel, H. (1997). Optimal robot excitation and identification. *IEEE Transactions on Robotics and Automation*, 13(5):730–740.

Takeyasu, K., Goto, T., and Inoyama, T. (1976). Precision insertion control robot and its application.

Taneja, I. J. (1989). On generalized information measures and their applications. In *Advances in Electronics and Electron Physics*, volume 76, pages 327–413. Elsevier.

Tang, T., Lin, H.-C., and Tomizuka, M. (2015). A learning-based framework for robot peg-hole-insertion. In *Dynamic Systems and Control Conference*, volume 57250, page V002T27A002. American Society of Mechanical Engineers.

Tao, G. (2003). *Adaptive control design and analysis*, volume 37. John Wiley & Sons.

Thompson-Butel, A. G., Lin, G. G., Shiner, C. T., and McNulty, P. A. (2014). Two common tests of dexterity can stratify upper limb motor function after stroke. *Neurorehabilitation and Neural Repair*, 28(8):788–796.

Tseng, P. and Yun, S. (2009). A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423.

Tura A., Lamberti C, Davalli A, S. R. (1998). Upper Limb Myoelectric Prosthesis With Cosmetic Covering. *Journal of rehabilitation research and development*, 35(1).

Valle, G., D'Anna, E., Strauss, I., Clemente, F., Granata, G., Di Iorio, R., Controzzi, M., Stieglitz, T., Rossini, P. M., Petrini, F. M., and Micera, S. (2020). Hand Control With Invasive Feedback Is Not Impaired by Increased Cognitive Load. *Frontiers in Bioengineering and Biotechnology*, 8.

Vantilt, J., Aertbeliën, E., De Groote, F., and De Schutter, J. (2015). Optimal excitation and identification of the dynamic model of robotic systems with compliant actuators. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2117–2124. IEEE.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and Contributors, S. . . (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Wen, Z., Shi, J., Li, Q., He, B., and Chen, J. (2018). ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:797–801.

Wilcox, G. and Nordstokke, D. (2022). Pediatric Co-Norms for Finger Tapping, Grip Strength, and Grooved Pegboard in a Community Sample. *Journal of the International Neuropsychological Society*, 28(1):85–93.

Williams, O. and Fitzgibbon, A. (2006). Gaussian process implicit surfaces. In *Gaussian Processes in Practice*.

Xia, P., Hu, J., and Peng, Y. (2018). EMG-Based Estimation of Limb Movement Using Deep Learning With Recurrent Convolutional Neural Networks. *Artificial Organs*, 42(5):E67–E77.

Xu, J., Koo, T.-K. J., and Li, Z. (2010). Sampling-based finger gaits planning for multifingered robotic hand. *Autonomous Robots*, 28(4):385–402.

Yi, Z., Calandra, R., Veiga, F., van Hoof, H., Hermans, T., Zhang, Y., and Peters, J. (2016). Active tactile object exploration with gaussian processes. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4925–4930. IEEE.

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9.

Zhang, Y., Ge, S. S., and Lee, T. H. (2004). A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2126–2132.

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. (2019). Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE.

Zhuang, K. Z., Sommer, N., Mendez, V., Aryan, S., Formento, E., D'Anna, E., Artoni, F., Petrini, F., Granata, G., Cannaviello, G., et al. (2019). Shared human–robot proportional control of a dexterous myoelectric prosthesis. *Nature Machine Intelligence*, 1(9):400–411.

Farshad KHADIVAR

Doctoral Assistant, Machine Learning & Robotics

Linkedin | Github

fredkvr@gmail.com
Chemin de Ruchoz 17,
1024 Ecublens, Switzerland
+41-78-3180055

Education

Ecole Polytechnique Fédérale de Lausanne (EPFL) Lausanne, Switzerland

Ph.D. Student, Robotics, Control, and Intelligent Systems - Advisor Prof. Aude Billard 2018 - Nov, 22 (Expected)

- Thesis: *"Advancing the Adaptability of Compliant Robot Controllers for Exploration, Interaction, and Manipulation"*
- Research focus: Supervised machine learning methods for complex manipulation tasks in human-centric environment

Sharif University of Technology Tehran, Iran

M.Sc. in Control and Mechatronics, GPA: 3.91/4, Ranked 1^{st} 2014 - 2017

- Thesis: *"Robust Model Predictive Controller To increase Transparency in Sinus Surgery Simulators"*
- Honors: *"Iran's National Elite Foundation Award"*, Sep. 2017

Languages

English (Proficient) **French** (Intermediate, B1 ongoing)

- English Teacher, with TEFL Certificate, Iran-Canada Institute of Languages, Tehran, Iran 2017 - 2018

Publications (first author)

Journal (*electronic version available upon request):

1. **Khadivar, F.**, Mendez, V., Batzianoulis, I., Correia, C., Micera, S. and Billard, A., 2022, under review*. *"Shared HumanRobot Compliant Control for In-Hand Manipulation with a Dexterous Myoelectric Prosthesis."* Journal of Neural Engineering.
2. **Khadivar, F.**, Chatzilygeroudis, K. and Billard, A., 2021 revised and resubmitted*. *"Self-Correcting Quadratic Programming-Based Robot Control."* IEEE Transactions on Systems, Man and Cybernetics: Systems.
3. **Khadivar, F.** and Billard, A., 2021, revised and resubmitted*. *"Adaptive Fingers Coordination for Robust Grasp and In-Hand Manipulation under Disturbances and Unknown Dynamics."* IEEE Transactions on Robotics (T-RO)
4. **Khadivar, F.**, Lauzana, I. and Billard, A., 2021. *"Learning dynamical systems with bifurcations."* Robotics and Autonomous Systems, 136, p.103700., DOI.

Conference:

6. **Khadivar, F.**, Gupta, S., Amanhoud, W. and Billard, A., 2021. *"Efficient Configuration Exploration in Inverse Dynamics Acquisition of Robotic Manipulators."* IEEE International Conference on Robotics and Automation (ICRA), pp. 1934-1941, DOI.
7. **Khadivar, F.**, Sadeghnejad, S., Moradi, H., Vossoughi, G. and Farahmand, F., 2017. *"Dynamic characterization of a parallel haptic device for application as an actuator in a surgery simulator."* RSI international conference on robotics and mechatronics (ICRoM) (pp. 186-191). IEEE.,DOI.

Skills

ML Domain:
- Supervised, and unsupervised learning
- Bayesian Learning, Gaussian Processes, Kernel Methods
- Mixture Models, Latent Variable Models
- Artificial Neural Networks, Reinforcement Learning
- Hidden Markov Models, Dirichlet Processes
- ML libraries and platforms: Pytorch, GPy, Limbo, Scikit-Learn

Robotics Domain:
- Control Theory (linear, nonlinear, robust, adaptive, optimal control)
- Robot Task Planning, Dynamical Systems
- Dexterous Manipulation, Adaptive Systems, Compliant Controllers
- Quadratic Programming, Learning from Demonstrations, Learning Dynamic Models

Programming: C++ | Python | Matlab | working knowledge of Bash

Software: ROS | Gazebo | DART | Motion Capture | Git | LaTeX| Office

Work Experience

Robotic Setups

- **Practical experience**: KUKA LBR IIWA, KUKA LWR, Allegro Hand.
- Worked with wearable sensors, motion capture system, and haptic devices.

Project Supervision at EPFL — 2019 - Ongoing

- **Master Thesis**: Sthithpragya Gupta (Spring-20) | **Master Projects**: Constantin Decaux (Spring-22), Raphael Uebersax (Fall-22), Bruno Agostinho Da Costa (Fall-22), Sascha Frey (Fall-19), Alexis Philip George-Georganopoulos (Fall-19)

Teaching Assistant, EPFL — 2019 - 2021

- "Applied Machine Learning," Prof. Aude Billard, (Fall-21 & Fall-19)

Academic Service — 2020 - Ongoing

- **Reviewing Journal**: RAL-2022 | **Reviewing Conference**: IROS-2020 to 2022 | ICRA-2021 | AAMAS-2021
- Administrator of Robotic Research Team, Amirkabir University of Technology, Tehran, Iran (2017 - 2018)