Human–Robot Interaction Based on Motion and Force Control

Martin Karlsson

LUND UNIVERSITY

Department of Automatic Control

PhD Thesis TFRT-1124
ISBN 978-91-7895-016-4 (print)
ISBN 978-91-7895-017-1 (web)
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

Industrial robots typically require detailed programming and carefully configured work cells to perform well. The large engineering effort implicates high cost and long preparation time, and this is the major obstruction when mediating tasks to robots. The research in this thesis therefore aims to make robot programming faster and more accessible. Methods that allow for programmers to mediate and modify tasks by means of demonstration are presented. Further, robots' abilities to replan with respect to unforeseen changes in their surroundings are enhanced, thus lowering the effort needed for work-cell configuration.

We first consider adjustment of robot movements generated by dynamical movement primitives (DMPs). DMPs are motion-control laws with emphasis on easy modification. For instance, goal configuration and time scale for a certain movement can be updated through one parameter each, commonly without further consideration. In this research, these capabilities are extended to support modifications based on demonstrations through physical human–robot interaction. Further, the motion-control laws are extended to support online replanning for overcoming unforeseen movement deviations.

Subsequently, a method that enables robots to recognize contact force/torque transients acting on the end-effector, without using a force/torque sensor, is proposed. This is achieved using machine learning. The robot is first exposed to examples of force/torque transients. Based on these data, a recurrent neural network (RNN) is trained to recognize such transients. The functionality is used to automatically determine when a robotic subtask is finished, to proceed to the next subtask at the right time. Finally, a control algorithm for teleoperation with force feedback is developed. It allows for an operator to demonstrate movement and forces remotely. One robot arm is moved directly through physical contact with the operator, and a distant robot arm moves accordingly. Interaction forces are reflected to each side of the interface.

Each of the methods presented in this thesis is implemented in a real-time application and verified experimentally on an industrial robot.

Acknowledgments

I would like to thank my supervisors Prof. Rolf Johansson and Prof. Anders Robertsson, for invaluable advice and support throughout this work. You have given me insights in all aspects of robot research, including theory, implementation, and presentation of results. Thanks to your ability to give me good advice in any situation, while also giving me the freedom to define my work, I have looked forward to every next day of this journey.

Prof. Bo Bernhardsson, I am happy that I took your advice to work at the department, both times. First as a teaching assistant, and then as a PhD student. I would also like to thank you and Prof. Fredrik Tufvesson again for the guidance as my Master's thesis supervisors many years ago.

Dr. Fredrik Bagge Carlson, it has been great to have you as a close colleague. It has been fun to improve robots together with you. We have a common interest not only in our work, but also in solid-state physics and especially defected semiconductors, and I look forward to review this area regularly together with you in the future.

Dr. Björn Olofsson, thank you for being a great role model and for generously sharing insights about robotics in both theory and practice. Your morale, good mood, and unquestionable competence set a great example.

Prof. Charlotta Johnsson, thank you for bringing me into the robotic surgery project Surgeon's Perspective. I really enjoy the work with you and our project colleagues, and I highly appreciate your guidance in where to direct our work. I would like to acknowledge M.D. Kiet Phan Tran for introducing me to cardiac surgery, for sharing your knowledge extremely generously, and for the interesting discussions we have about the intersection of robotics and surgery.

I would like to acknowledge my colleagues and former colleagues in the Robotics Lab, Dr. Mahdi Ghazaei Ardakani, Dr. Anders Nilsson, Anders Blomdell, Pontus Andersson, Asst. Prof. Mathias Haage, Prof. Jacek Malec, Assoc. Prof. Elin Anna Topp, Assoc. Prof. Klas Nilsson, Dr. Maj Stenmark, Dr. Magnus Linderoth, Dr. Olof Sörnmo, Martin Holmstrand, Matthias

Mayr, Prof. Volker Krüger, Julian Salt, Alexander Dürr, Dr. Karl Berntorp, and Dr. Andreas Stolt. Thank you for great cooperation.

Dr. Mårten Wadenbäck, thank you for introducing me to the exciting field of visual odometry, and for being a great collaborator and friend. It was worth all the difficulties with network connections, camera settings, and even two minor electric shocks.

Throughout the years I have had the honor and pleasure of sharing office with many great colleagues. Thank you Olof Troeng, Dr. Björn Olofsson, Nils Vreman, Martin Morin, Victor Millnert, Dr. Ola Johnsson, Lic. Tech. Jonas Dürango, Irene Zorzan, Dr. Meike Rönn, Dr. Magnus Linderoth, Dr. Gabriel Turesson, and Lic. Tech. Alina Andersson, for creating such a nice working climate!

Prof. Karl Johan Åström, you are gratefully acknowledged for fruitful discussions about this thesis, and about automatic control in general. You are a great source of knowledge and inspiration at our department.

I am grateful to the administrative staff at the department, Ingrid Nilsson, Mika Nishimura, Monika Rasmusson, Cecilia Edelborg Christensen, and Dr. Eva Westin, for your help with various things related to work. You are always so positive, well organized, and helpful. A special thanks to Ingrid and Monika, for always being supportive and understanding when I did my best with project-time reporting.

Leif Andersson, thank you for helping me using LaTeX to its full potential. Without your aid, this thesis, as well as my other publications, would have looked much worse. You have helped me to position figures correctly, number equations and subequations, and to remove unnecessary LaTeX code that I naively copied from the Internet. On a related note, I thank Dr. Björn Olofsson, Axel Karlsson, Dr. Fredrik Bagge Carlson, Julian Salt, Matthias Mayr, and Dr. Mahdi Ghazaei Ardakani for proofreading of this thesis.

I also thank all colleagues at the department. Each one creates a positive, inspiring, and fun working environment, and contributes to interesting discussions about control theory and related topics.

Last but not least, Daniel, Ann-Eli, Axel, and Johan Karlsson, have always been a fantastic family. Heartfelt thanks for all the love and support!

Financial Support

Financial support is gratefully acknowledged from the VINNOVA project *Kirurgens Perspektiv* (Surgeon's Perspective in English). Financial support is also gratefully acknowledged from the European Commission, under the Framework Programme Horizon 2020 within grant agreement No 644938–SARAFun, and under the 7th Framework Programme within grant agreement No 606156–FlexiFab, as well as from the Swedish Foundation for Strategic Research through the SSF project ENGROSS. The author is a member of the LCCC Linnaeus Center, supported by the Swedish Research Council, and the ELLIIT Excellence Center, supported by the Swedish Government.

Contents

1

Introduction

This thesis considers human–robot interaction, both for robot programming and for collaboration during program execution. The overall aim is to make robot programming more accessible. The research is based on motion control, force control, and machine learning. Motion-control algorithms that generate robot movement capable of adapting to unforeseen events are developed. Force control is used to allow for a programmer to demonstrate desired movement and expected interaction forces. Further, machine learning is used to recognize interaction forces between robot and work object, in order to detect key events during robotic manipulation. Such detection can, for instance, be used to determine when a subtask has been accomplished, in order to switch to the next subtask in the program.

1.1 Background and Motivation

The long-term ambition in the field of robotics is broad: Robots should do work to enrich humanity. The word robot comes from *robota*, the Czech word for work. In some cases, we would like robots to replace human workers. There might be several reasons for certain tasks to be undesirable for humans. Such tasks might be too time consuming, monotonous, unhealthy, or even hazardous. In other cases, we would like robots to do work that can not be done by humans, for instance due to physical limitations. Already today, robots are used in a wide range of tasks. The automotive industry is extensively robotized, and robots for vacuum cleaning and lawn mowing are common in households. Robots are also used in surgery and planet exploration, but in contrast to the previous examples, these are controlled remotely by humans instead of operating completely autonomously; see, *e.g.*, [Spong et al., 2006; Siciliano et al., 2010] for an introduction to robot modeling and control.

Industrial robots typically operate by performing sequences of predefined movements, with high speed, high position accuracy, and without rest. This has already enabled automation of repetitive tasks where position control suffices, such as spray painting and welding. For such tasks, robots can be programmed to outperform humans.

However, predefined robotic movements are meaningful only in carefully structured environments, specifically designed to fit the robot and the given task. The task must therefore be highly repetitive, and possible deviations from the original plan must have been foreseen by the robot programmer. Even under favorable conditions, traditional robot programming is time consuming and requires expert knowledge. As a result, human labor is still more cost effective than automation for many seemingly repetitive tasks, such as assembly tasks. Electronic products such as smartphones and robot vacuum cleaners are put together manually in large factories. Unfortunately, these assembly tasks are too repetitive to be desirable for humans, and at the same time insufficiently repetitive to be robotized. Robotic assembly has been addressed in, *e.g.*, [Thomas and Wahl, 2001; Thomas et al., 2007; Björkelund et al., 2011; Stolt et al., 2012a; Linderoth, 2013; Stolt, 2015]. For human workers, these assembly tasks cause fatigue and leave little room for personal development. Even though these tasks appear monotonous for humans, small tolerances and tiny variations between similar assembly parts make it inadequate to just perform a series of accurate movements. There is also a trend toward manufacturing a given product in a smaller volume and for a shorter time, and then changing to a new one. This shortens the acceptable preparation time even further.

Industrial robots are commonly strong and heavy, and can not safely operate close to humans. There is nowadays a trend to build inherently safe robots, with less weight and less powerful motors. YuMi [ABB Robotics, 2018], Panda [Franka Emika, 2018], Baxter [Rethink Robotics, 2018], and LBR iiwa [KUKA, 2018], are examples of such collaborative robots, also referred to as light-weight robots. This type of safe hardware is appropriate for the physical human–robot interaction considered in this thesis.

Many tasks that humans do in society, such as buying groceries, cooking, and cleaning, are not yet possible to fully automate. Even though dedicated machines such as dish washers and vacuum cleaners exist, significant human engagement is still required to make these machines useful. Predefined movement sequences would by no means work in an everyday environment, because it is not sufficiently predictable. Instead, a general ability to adapt to the surroundings is necessary. This requires a more general type of intelligence than robots currently possess.

These circumstances have motivated the following research objectives:

1. Enable faster and more intuitive robot programming;
2. Enable robots to take suitable action with respect to their tasks and surroundings.

In this thesis, research toward these objectives is presented. The objectives are partly overlapping. For example, if a robot can replan with respect to its work space, the programmer does not have to consider all possible scenarios in advance. This reduces the required programming competence and effort. Vice versa, intuitive means of robot programming could allow for an operator to mediate suitable behavior, given certain states or events, to the robot, and thereby enhance the adaptability.

The research in this thesis has been done within the projects SARAFun [SARAFun, 2019] and Surgeon's Perspective. SARAFun is short for Smart Assembly Robot with Advanced Functionalities, and the aim of the project was to enable non-expert operators to program collaborative robots for assembly tasks. SARAFun was concluded at a final review in April 2018, where some of the research in this thesis was presented and demonstrated. The aim of Surgeon's Perspective is to develop robot-assisted surgery, and the thesis author has been responsible for developing the algorithms for robot control described in this thesis. The two projects overlap, as both aim to facilitate robot programming and enhance online replanning capabilities.

1.2 Robot Programming and Control

Whereas humans would prefer to program on a high level of abstraction, for instance through natural language, robots require very detailed instructions, for instance time series of desired joint torques. Robot programming is therefore difficult in general. High-level instructions can work well between humans, but leave too many details unspecified to be suitable for robot programming.

For a robot program to be realizable, it is necessary that appropriate low-level quantities such as motor currents can be determined automatically and unambiguously based on the program and on measurements or estimates. Thanks to automatic control [Wen and Murphy, 1990; Wen and Murphy, 1991; Åström and Hägglund, 1995; Khalil, 2002; Åström and Wittenmark, 2013], of which basic knowledge is assumed in the following, robot programs can be specified on more intuitive levels than that of motor currents. Some applications require only that the robot reaches desired positions. It is then sufficient to specify these positions. Motion control is used to determine the corresponding joint torques in real time. In turn, inner control loops are used to command motor currents that realize the joint torques. Similar to motion control, desired interaction forces at the robot end-effector can be achieved using force control, though this introduces additional challenges as addressed in [Johansson et al., 2015].

Robot programming may consist of ordinary computer programming, disregarding hardware considerations. Source code is written on a computer and automatically transformed to executable machine code. To facilitate the programming, graphical user interfaces such as ABB RobotStudio [ABB Robotics, 2019b] have been introduced. Reference coordinate systems and sequences of movement targets can be defined graphically and converted to source code. It is also common to move the physical robot with a joystick to desired configurations, and save these as part of the code. Still, many operations are more suitable to define by direct source-code editing. Further, programs can be tested with visual simulation tools such as RobotStudio, rviz [ROS, 2018], and Gazebo [Gazebo, 2018]. These established means of robot programming and simulation require special knowledge.

An easier way to mediate behavior is by means of demonstration. Robot programming by demonstration is also referred to as robot learning from demonstration, apprenticeship learning, and imitation learning. A survey is given in [Argall et al., 2009]. Finding robust and general frameworks for programming by demonstration is a current research challenge. One possibility is to let the user do the task, document it by, *e.g.*, video recording, and automatically generate a robot program that achieves the same thing. However, in that kind of procedure it is difficult for the user to take all physical limitations of the robot, such as gripper design, joint limits, and reach limits, into account. A natural way to incorporate these limitations is to instead let the user guide the robot through the intended task, for instance by means of lead-through programming where the robot is moved through physical contact with the user [Pan et al., 2010; Stolt et al., 2015a; Capurso et al., 2017]. The intended configurations of the robot will then be unambiguously demonstrated, at least given the work-space configuration. This is the kind of robot programming considered in this thesis. Demonstrated movement can be represented and replayed by, *e.g.*, dynamical movement primitives (DMPs), see [Ijspeert et al., 2013] and Chapter 3. Further, longer demonstrations can be segmented automatically into key phases using, *e.g.*, probabilistic approaches [Fox et al., 2009; Lee et al., 2015].

Not only configuration data can be mediated through demonstration. For instance, image data, force/torque data, *etc.*, at different stages of a task can be mediated to the robot in the form of examples. Based on these examples, model parameters can be determined automatically using machine learning [Bishop, 2007; Murphy, 2012] or deep learning [Goodfellow et al., 2016]. Such models could potentially be used by the robot to autonomously monitor the task.

1.3 Introduction to Machine Learning

Computers and robots have traditionally been programmed by writing explicit code, specifying sets of rules and behaviors in detail. In predictable scenarios this strategy works well, but most of the tasks that humans accomplish in their everyday life are far too complex to mediate in such fashion. Whereas distinguishing, for instance, whether a certain image represents a car or a bicycle is typically easy for humans, hand crafting the classification rules from raw image data is not feasible. The introduction in [Goodfellow et al., 2016] provides interesting examples of tasks that are easy for humans to perform, but difficult to describe formally.

Machine learning is the most promising framework to address such problems. This framework consists of three major parts; supervised learning, unsupervised learning, and reinforcement learning [Bishop, 2007; Murphy, 2012; Goodfellow et al., 2016], all of which are relevant in robotics. The main idea is to provide computers with example data, and use these to define models, instead of specifying the models manually. In this thesis, we mainly consider supervised learning. Whereas traditional automatic control is useful for achieving specific robot movements, machine learning can potentially allow for programming on higher levels of abstraction. For instance, if natural language or image data could be interpreted in a meaningful way, these could potentially be used as a decision basis by a robot. In machine learning, such data are typically analyzed using artificial neural networks [Kröse and Smagt, 1993; Goodfellow et al., 2016], which were inspired by the connected neurons in biological brains.

Iterative learning control is closely related to reinforcement learning, and has been used for motion control [Norrlöf, 2002; Norrlöf and Gunnarsson, 2002] and force control [Marchal et al., 2014] of industrial robots. The main idea is to iteratively update the control signal based on previous trials, to compensate for repetitive sources of error. System identification [Ljung, 1987; Johansson, 1993; Lindsten et al., 2013; Schön et al., 2015] is closely related to machine learning, and is specialized on finding dynamical models from time series of input and output signals.

Supervised machine learning is used to approximate a given function, $y(x)$, with a parameterized function, $\hat{y}(x|\theta)$, which takes some input data x, and maps it to an output $\hat{y}$. Here, θ denotes the model parameters. In the example of image recognition, x could be pixel values, y would be the true image category, and $\hat{y}(x)$ could be interpreted as the probability distribution over the two categories, *i.e.*, car and bicycle, given x.

In order to learn $\hat{y}(x)$, the model is exposed to a large data set of examples, called training data. In supervised learning, the training data consist of both input data and the corresponding known outputs, usually manually labeled. In the training phase, the elements of θ are adjusted to fit the training data by means of optimization. A loss function, L, in which some measure of the error of $\hat{y}(x)$ compared to $y(x)$ is included, is minimized with respect to the model parameters.

Since the training data can only include a small subset of all possible data points, an important aspect of machine learning is generalization, *i.e.*, to predict the output given input not used during training. In order to achieve this, the complexity of the model is typically restricted, by keeping the number of parameters low, or by penalizing the complexity by including some measure of it in L. Further, test data, not directly used to optimize the model parameters, are used to estimate how well the model generalizes. It is, however, common to determine some model hyperparameters based on the performance on test data. Therefore, it is good practice to use yet another data set to investigate the generalizability, without affecting the model in any way. Such data are called validation data.

This general approach is adopted in Chapter 7, where the aim is to take a step toward more accessible robot programming. Ideally, a non-expert operator should be able to provide a robot with data, enabling it to learn from experience. Similar to the image-classification example, a model is trained to determine a class given input data. In particular, the input consists of robot joint torques, and the task is to determine whether a certain force/torque transient, acting on the robot end-effector, is present or not. One important difference from the image-recognition example is that the input consists of a time series rather than a static representation, which should be taken into account when choosing the structure of the model. Related approaches have been presented in [Rodriguez et al., 2010; Rojas et al., 2012; Stolt et al., 2015b].

1.4 Problem Formulation

This section describes the problems addressed on a conceptual level. More technical problem formulations are given gradually throughout this thesis. In summary, the main problem addressed in this thesis is that robot programming is expensive and time consuming, and that robots, once programmed, exhibit low adaptability to the environment.

The first aim of this thesis is to investigate whether it is possible to automatically interpret demonstrated corrections of robotic movement. In the scenarios addressed, the original movement has a faulty ending, for instance as a result of new work-space configuration. It would be desirable to specify the correction through demonstration, and the result of the correction should be sufficiently predictable for an operator.

A common problem is robots' inability to handle unforeseen events. Demonstrated movements can be saved and replayed exactly, but such functionality is meaningful only in predictable environments. Unforeseen events may force a robot to deviate from the intended movement, for instance to avoid collision. It should therefore be investigated whether unforeseen deviations of robotic movement could be recovered from, so that the intended task could still be accomplished. A motion controller to achieve this should be formulated. Such a control system should have the necessary features known from automatic control: asymptotic stability, safe stability margins, and control signals of moderate magnitude. It should support both real coordinate spaces such as position in joint or Cartesian space, and orientation in Cartesian space which is more challenging due to the topology of the 3D rotation group SO(3).

Robotic manipulation requires in each step a validation procedure, to ensure that the intended subtask has been accomplished. Without such validation, successive subtasks and eventually the entire manipulation task would be jeopardized. This could leave the task unfinished, and even damage the robot or the work objects. We therefore address the question of whether validation models could be mediated to a robot through demonstration, rather than explicit programming which is nowadays the most common approach.

In ordinary lead-through programming, the operator demonstrates movements by moving the robot arm through physical contact in such a way that the robot arm will complete an intended task. This strategy suffices to teach behavior in the position domain. Contact forces induced between the robot and the work objects are also of importance, but more difficult to demonstrate. Unfortunately, these contact forces can not be distinguished from contact forces from the operator in ordinary lead-through programming. Finally, we therefore address the question of whether the contacts could be separated using two robot arms, where one is in contact with the operator allowing for direct lead-through programming, and the other one is controlled remotely to manipulate the work object.

1.5 Contributions

The main contributions of this thesis are:

- A control architecture for adjustment of robot movement through physical human–robot interaction;
- Improved replanning capabilities of robot movements;
- Sensorless detection of contact-force transients;
- Implementation of a dual-arm haptic interface for task demonstration.

Each contribution includes experimental validation on an industrial robot.

1.6 Publications

In this section, publications authored or co-authored by the thesis author are presented. The published manuscripts have been subjected to peer review. For each manuscript, A. Robertsson and R. Johansson have contributed with supervision, including insights regarding previous work and our ongoing research, as well as suggestions for improvement. First, the publications on which this thesis is based are described.

Karlsson, M. (2017). *On Motion Control and Machine Learning for Robotic Assembly*. Licentiate Thesis, TFRT–3274–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Parts of the research presented in this PhD thesis have previously been published in the Licentiate thesis above.

Karlsson, M., A. Robertsson, and R. Johansson (2017). "Autonomous interpretation of demonstrations for modification of dynamical movement primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 29–June 3, Singapore, pp. 316–321.

In the publication above, M. Karlsson formulated the method for updating a partly faulty trajectory representation, based on a corrective demonstration. Further, M. Karlsson implemented the method and verified it experimentally. Chapter 4 is based on this publication. By this we mean that the chapter contains an updated version of the publication, where the author has made changes and extensions as found suitable.

Karlsson, M., F. Bagge Carlson, A. Robertsson, and R. Johansson (2017). "Two-degree-of-freedom control for trajectory tracking and perturbation recovery during execution of dynamical movement primitives". In: *20th IFAC World Congress*. July 9–14, Toulouse, France, pp. 1959–1966.

M. Karlsson and F. Bagge Carlson identified the necessity of augmenting the existing DMP framework, to make related research approaches on DMP perturbation recovery practically realizable. M. Karlsson formulated the augmentation, and verified it in simulations and experimentally, while frequently discussing the work with F. Bagge Carlson. Further, F. Bagge Carlson implemented the method presented as an open-source Julia package, which can be found online [Bagge Carlson, 2016]. Example code in Matlab, written by M. Karlsson, can be found online [Karlsson, 2017c]. Chapter 5 is partly based on this publication.

Karlsson, M., A. Robertsson, and R. Johansson (2018). “Convergence of dynamical movement primitives with temporal coupling”. In: *European Control Conference (ECC)*. June 12–15, Limassol, Cyprus, pp. 32–39.

M. Karlsson formulated the convergence proof and verified it experimentally. Chapter 5 is partly based on this publication.

Karlsson, M., A. Robertsson, and R. Johansson (2019). “Temporally coupled dynamical movement primitives in Cartesian space”. Manuscript prepared for submission to review for publication.

M. Karlsson formulated the control algorithm and verified it experimentally. Chapter 6 is based on this manuscript.

Karlsson, M., A. Robertsson, and R. Johansson (2018). “Detection and control of contact force transients in robotic manipulation without a force sensor”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 21–25, Brisbane, Australia, pp. 21–25.

M. Karlsson formulated the transient-detection method and verified it experimentally. Chapter 7 is based on this publication.

Ghazaei Ardakani, M. M., M. Karlsson, K. Nilsson, A. Robertsson, and R. Johansson (2018). “Master-slave coordination using virtual constraints for a redundant dual-arm haptic interface”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. October 1–5, Madrid, Spain, pp. 8751–8757.

The idea of the dual-arm haptic interface was conceived by K. Nilsson and M. Ghazaei Ardakani. The latter developed and formulated the control algorithm for the haptic interface. M. Karlsson implemented the interface and evaluated it experimentally. Chapter 8 is based on this publication. Andreas Stolt, Cognibotics AB, is gratefully acknowledged for identifying the dynamics of the robot used in the experiments. Mathias Haage at Dept. Computer Science, Lund University, is gratefully acknowledged for guidance in compiling separate parts of the software implementation with different compilation flags.

The following publications cover topics in robotics, nonlinear state estimation, and positioning. They are, however, outside the scope of this thesis.

Bagge Carlson, F., M. Karlsson, A. Robertsson, and R. Johansson (2016). “Particle filter framework for 6D seam tracking under large external forces using 2D laser sensors”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. October 9–14, Daejeon, South Korea, pp. 3728–3734.

Haage, M., S. Profanter, I. Kessler, A. Perzylo, N. Somani, O. Sörnmo, M. Karlsson, S. G. Robertz, K. Nilsson, L. Resch, and M. Marti (2016). “On cognitive robot woodworking in SMErobotics”. In: *ISR 2016: 47th International Symposium on Robotics*. VDE. June 21–22, Munich, Germany, pp. 1–7.

Karlsson, F., M. Karlsson, B. Bernhardsson, F. Tufvesson, and M. Persson (2015). “Sensor fused indoor positioning using dual band WiFi signal measurements”. In: *European Control Conference (ECC)*. July 15–17, Linz, Austria, pp. 1669–1672.

Karlsson, M., F. Bagge Carlson, J. De Backer, M. Holmstrand, A. Robertsson, and R. Johansson (2016). “Robotic seam tracking for friction stir welding under large contact forces”. In: *7th Swedish Production Symposium (SPS)*. October 25–27, Lund, Sweden.

Karlsson, M. and F. Karlsson (2016). “Cooperative indoor positioning by exchange of bluetooth signals and state estimates between users”. In: *European Control Conference (ECC)*. June 29–July 1, Aalborg, Denmark, pp. 1440–1444.

Persson, M., M. Karlsson, and F. Karlsson (2014). *Method for improved indoor positioning and crowd sourcing using PDR*. International patent, Publication number: WO 2015/118369 (A1), Applicant: Sony Corporation, Examined and granted under the patent cooperation treaty (PCT). Sony. URL: https://patents.google.com/patent/WO2015118369A1/en (visited on 2019-03-02).

Wadenbäck, M., M. Karlsson, A. Heyden, A. Robertsson, and R. Johansson (2017). “Visual odometry from two point correspondences and initial automatic camera tilt calibration”. In: *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Volume 6*. VISIGRAPP. February 27–March 1, Porto, Portugal, pp. 340–346.

1.7 Thesis Outline

This thesis is a monograph. In terms of scientific contributions, Chapters 4 to 6 focus on movement adjustment using motion control, whereas Chapters 7 and 8 focus on programming by demonstration based on machine learning and force control. Despite this division, both programming by demonstration, motion control, force control, and machine learning are used throughout the thesis to put the contributions into context. Finally, a conclusion is presented in Chapter 9. References in the digital version of the thesis are clickable and marked in blue.

Chapter 2: Robots and Interfaces for Experimental Validation

The robots and research interfaces used in the experiments in this thesis are presented.

Chapter 3: Dynamical Movement Primitives—An Overview

The fundamentals of DMPs, used to model and generate robot movement, have been developed in previous research. In this chapter, the main results from previous research regarding DMPs are presented. Further, the DMP framework is compared with alternative movement models.

Chapter 4: Modification of Robot Movement Using Lead-Through Programming

A framework for modification of DMPs by means of corrective demonstrations is presented. It allows for an operator to adjust a faulty part of a trajectory using lead-through programming, while retaining satisfactory parts.

Chapter 5: Temporally Coupled Dynamical Movement Primitives in $\mathbb{R}^n$

The DMP framework is augmented to enable recovery from perturbations during DMP execution. The control algorithm for temporal coupling proposed in [Ijspeert et al., 2013] is used as a foundation, and evaluated in the beginning of this chapter. It is first shown that it is necessary to lower the feedback control gains in the previously proposed algorithm. It is further shown that it is necessary to add a feedforward signal to the controller. A new control algorithm for temporally coupled DMPs is proposed based on these insights. It is shown mathematically that exponential convergence to the goal state can be guaranteed. The configuration space is assumed to be a real coordinate space, such as joint space or position in operational space.

Chapter 6: Temporally Coupled Dynamical Movement Primitives in 3D Orientation Space

The temporally coupled DMPs are extended to support orientation in Cartesian space. Orientations are represented by unit quaternions, and the special topology of the unit quaternion set is taken into account in the design and stability analysis of the control system.

Chapter 7: Detection and Control of Contact-Force Transients

A machine-learning procedure for detecting force/torque transients acting on a robot end-effector, by measuring robot joint torques, is developed. The procedure does not require any external force/torque sensor, the optimization time of the detection model is moderate, and the detection model can generalize to other tasks not used for model training. These are the main benefits as compared to the state of the art.

Chapter 8: A Dual-Arm Haptic Interface for Task Demonstration

An immersive haptic interface for task teaching is proposed. By retaining a fixed pose offset between two robot end-effectors in task space, while enabling lead-through programming on both arms, any arm can be controlled directly by an operator while the other one moves accordingly. Interaction forces are reflected to each side as haptic feedback. Hence, the operator can act and sense through the system. The interface supports mapping between two not-necessarily-similar robot arms, mounted on any surfaces and with any initial configurations. Further, the arms can be interacted with on any point on their structures.

1.8 Videos

Videos that show each of the functionalities developed are available in the entry for this thesis in the Lund University Research Portal:
`http://portal.research.lu.se/portal/`

These videos are also available on the author's Youtube channel:
`https://www.youtube.com/channel/UCIYU9qpJwkBbsFWscI2vV-g`

Each video will be referred to and explained throughout this thesis.

2

Robots and Interfaces for Experimental Validation

Most of the experiments presented in this thesis have been done in the Robotics Lab shared by the departments of Automatic Control and Computer Science at Lund University, and some have been done at ABB Corporate Research in Västerås, Sweden. Different versions of the ABB YuMi robot have been used. YuMi is a dual-arm collaborative robot [Kock et al., 2011; ABB Robotics, 2018]. Each arm has seven joints and is therefore redundant. It is designed for safe human–robot interaction [Matthias et al., 2011], with low mass, weak motors, smooth surfaces, and speed limitations. Each joint is controlled by the ABB IRC5 control system [ABB Robotics, 2019a], running at 2 kHz. The control system consists of a current-control loop to achieve desired joint torque, and outer control loops to achieve desired joint position and velocity; see, *e.g.*, Fig. 2.4 in [Stolt, 2015] for a block diagram of the IRC5 system. In this thesis, we will refer to the IRC5 system as the internal robot controller. The algorithms developed in this thesis were implemented on ordinary PCs, and communicated with the internal robot controller through the research interfaces described in the following sections.

YuMi is backdrivable, which means that it is possible to move it through physical contact without being prevented by high friction, large mass, and high gear ratio between the arm and motor sides. This is a common property for light-weight robots, whereas larger traditional industrial robots are typically not backdrivable. Sensorless lead-through programming is easier to achieve for robots with such backdrivability. In [Stolt et al., 2015a], it was implemented on the YuMi prototype described in Sec. 2.1. A version of that implementation is included in the YuMi product (see Sec. 2.2) by default.

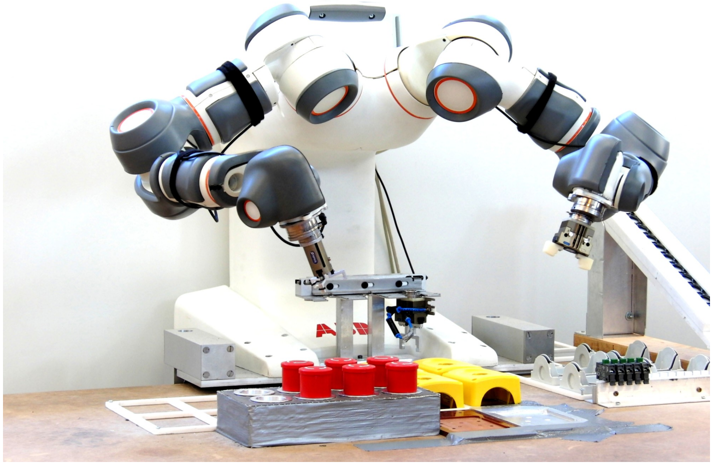


Figure 2.1 The prototype version of YuMi [ABB Robotics, 2018] used in the experiments. It was located in the Robotics Lab at Lund University.

2.1 YuMi Prototype Version

The prototype version of YuMi used in this thesis is shown in Fig. 2.1. The research interface ExtCtrl [Blomdell et al., 2005; Blomdell et al., 2010], running at 250 Hz, was used for communication between the PC and the internal robot controller. Each wrist of the robot was equipped with a six-degree-of-freedom ATI [ATI, 2019] Mini40 force/torque sensor. In this thesis, these external sensors were used for validation only.

2.2 YuMi Product Version

The product version of YuMi mainly used in this thesis was located in the Robotics Lab at Lund University, and is shown in Fig. 2.2. Some of the experiments in Chapter 7 were done at ABB Corporate Research in Västerås, on the robot shown in Fig. 2.3. This robot was also used for some separate demonstrations in the SARAFun project. The main demonstration platform used in SARAFun is shown in Fig. 2.4. The product version of YuMi was controlled through a research-interface version [Bagge Carlson and Haage, 2017] of Externally Guided Motion (EGM) [ABB Robotics, 2019c], running at 250 Hz. The product version is not equipped with force/torque sensors by default.

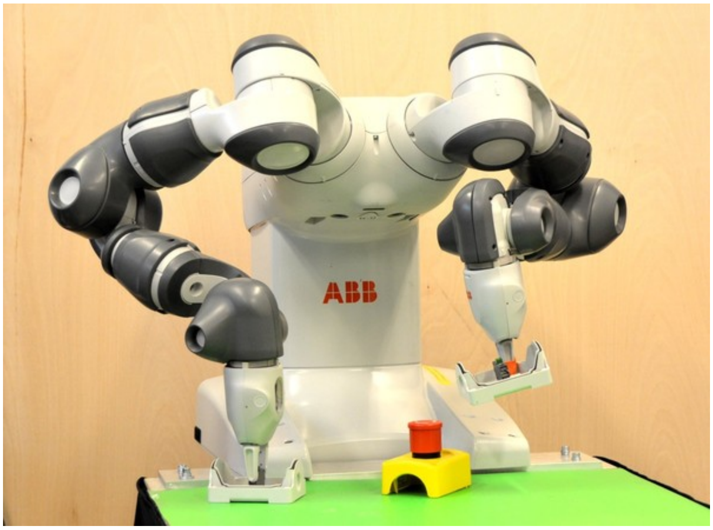


Figure 2.2 The product version of YuMi mainly used in the experiments. It was located in the Robotics Lab at Lund University.

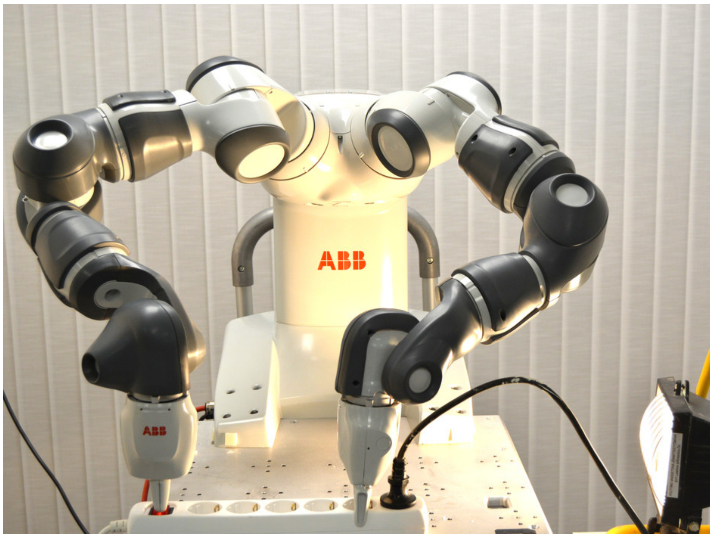


Figure 2.3 The product version of YuMi located at ABB Corporate Research in Västerås, used for some of the experiments in Chapter 7 and for some separate demonstrations in SARAFun.

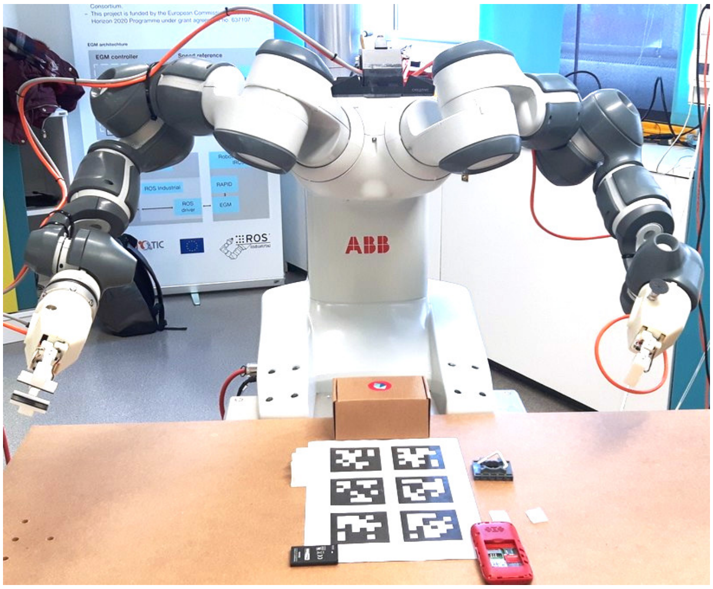


Figure 2.4 The product version of YuMi located at ABB Corporate Research in Västerås, used as main demonstration platform in SARAFun.

3

Dynamical Movement Primitives—An Overview

Since robot programs consist of sequences of movements, representation and execution of these movements is an important area in robotics. In industrial settings, it is necessary that the movements can be realized according to plan. To enhance robots' replanning capabilities, dynamical movement primitives (DMPs) have been developed. These are used to represent and execute robot movement, and have been developed with programming by demonstration in mind. We consider the work in [Ijspeert et al., 2013] as the main source from previous research, because it presents a complete description of the framework together with some possible extensions and applications. This chapter presents an overview of DMPs, but no extension of the framework as compared to [Ijspeert et al., 2013]. Earlier versions have been introduced in [Schaal et al., 2000; Ijspeert et al., 2002; Schaal et al., 2003; Ijspeert et al., 2003]. The framework has been inspired by the biological movement models presented in [Giszter et al., 1993; Mussa-Ivaldi, 1999]. Table 3.1 lists some of the notation used in this chapter.

Although there are other alternatives, the most ubiquitous model for DMPs is based on the following dynamical system.

$$\tau^2 \ddot{y}_{\mathrm{r}} = \alpha(\beta(g - y) - \tau \dot{y}) + f(x) \tag{3.1}$$

Here, y denotes robot configuration, α and β are positive constants, and $f(x)$ is a so-called forcing term where x is a phase variable. There are three aspects that support movement modification: The speed of the system can be set through the positive time parameter τ, the goal configuration can be set through g, and movement can be generated from any state. Because of $f(x)$, the system is nonlinear. It is commonly pointed out that (3.1) represents a damped-spring system, where $f(x)$ can be seen as an external force that directly affects the acceleration [Ijspeert et al., 2013]. In this thesis, we will view DMPs more from an automatic-control perspective,

Table 3.1 Notation used in this chapter.

Notation		Description
n	$\in \mathbb{Z}^+$	Dimension of robot configuration
y	$\in \mathbb{R}^n$	Robot configuration
$\ddot{y}_\mathrm{r}$	$\in \mathbb{R}^n$	Robot acceleration reference
τ	$\in \mathbb{R}^+$	Time parameter
g	$\in \mathbb{R}^n$	Goal state
x	$\in \mathbb{R}^+$	Phase variable
$f(x)$	$\in \mathbb{R}^n$	Learnable virtual forcing term
α, β, α_x	$\in \mathbb{R}^+$	Positive constants
N_b	$\in \mathbb{Z}^+$	Number of basis functions
$\Psi_j(x)$	$\in \mathbb{R}^n$	The j:th basis function vector
w_j	$\in \mathbb{R}^n$	The j:th weight vector

while retaining the notation from [Ijspeert et al., 2013] where applicable. The system consists of a proportional-derivative (PD) controller, together with feedforward by $f(x)$. The dynamics are achieved by sending $\ddot{y}_\mathrm{r}$ as a reference acceleration to internal motion-control loops. It is assumed that the reference acceleration can be realized during free-space motion, so that $\ddot{y} = \ddot{y}_\mathrm{r}$. This is reasonable for an acceleration of moderate magnitude and time derivative. Depending on the level of control, $\ddot{y}_\mathrm{r}$ can also be seen as a control signal. The control structure is visualized in Fig. 3.1.

Consider for now the simplified case where $f(x) = 0$, so that

$$\tau^2 \ddot{y}_\mathrm{r} = \alpha(\beta(g - y) - \tau\dot{y}) \tag{3.2}$$

This is a PD controller with g as reference configuration. In the context of robot position control, overshoot is undesirable due to the risk of collision.

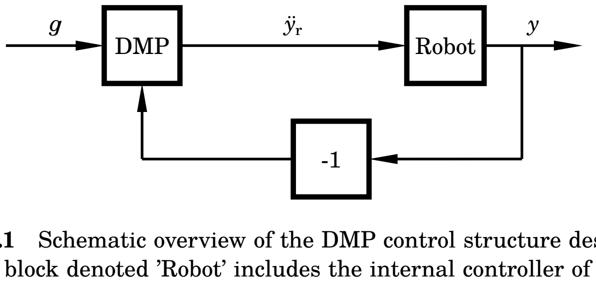


Figure 3.1 Schematic overview of the DMP control structure described by (3.1). The block denoted 'Robot' includes the internal controller of the robot. The block denoted 'DMP' can be seen as a controller, that generates $\ddot{y}_\mathrm{r}$ as control signal. It can also be seen as a reference generator, or planner, that generates $\ddot{y}_\mathrm{r}$ as reference for the internal robot controller.

This is especially important in operations that involve contact with the environment. This fact motivates the absence of an integral part, which is otherwise common in control applications [Åström and Hägglund, 1995] but may cause overshoot. Further, it is advisable to choose the feedback control parameters α and β so that the system is critically damped, *i.e.*, so that y converges asymptotically to g without overshoot. In [Ijspeert et al., 2013], it was stated that this is achieved for $\beta = \alpha/4$. In the following, we will verify this by analyzing the poles of the system. This can be done separately for each dimension of the configuration, and to keep the notation uncluttered we consider a one-dimensional case. Note that the dynamical system (3.2) is linear. The transfer function $G_{g\to y}(s)$ for the closed-loop system in Fig. 3.1 with $f(x) = 0$ is given by

$$\frac{Y(s)}{G(s)} = G_{g\to y}(s) = \frac{\dfrac{\alpha\beta}{\tau^2}}{s^2 + \dfrac{\alpha}{\tau}s + \dfrac{\alpha\beta}{\tau^2}} \tag{3.3}$$

Here, $Y(s)$ represents the Laplace transform of y, and $G(s)$ is the Laplace transform of g. Inserting $\beta = \alpha/4$ yields

$$\frac{Y(s)}{G(s)} = G_{g\to y}(s) = \frac{\dfrac{\alpha^2}{4\tau^2}}{\left(s + \dfrac{\alpha}{2\tau}\right)^2} \tag{3.4}$$

Both poles are located in $-\alpha/(2\tau)$. Because these are strictly negative, global asymptotic stability can be concluded. Because they are real, no overshoot will occur. We also see the effect of τ. A larger value yields poles closer to the origin, and hence slower system dynamics.

The controller in (3.2) hence drives the configuration to the desired state g, and the speed of the system can be adjusted. While this is satisfactory for a wide range of control applications, it is in robot motion control very important which trajectory is traversed to reach a desired end point. For instance, reach limits, joint angle limits, and external obstacles must be avoided. The purpose of $f(x)$ in (3.1) is to enable specification of the trajectory traversed. Each element of $f(x)$, denoted $f_i(x)$, is given by a weighted sum of radial basis functions

$$f_i(x) = \frac{\sum_{j=1}^{N_\mathrm{b}} \Psi_{i,j}(x) w_{i,j}}{\sum_{j=1}^{N_\mathrm{b}} \Psi_{i,j}(x)} x \cdot (g_i - y_{0,i}) \tag{3.5}$$

where each basis function, $\Psi_{i,j}(x)$, is determined as

$$\Psi_{i,j}(x) = \exp\left(-\frac{1}{2\sigma_{i,j}^2}(x - c_{i,j})^2\right) \tag{3.6}$$

Here, N_b denotes the number of basis functions, w denotes weights, y_0 is the initial configuration, and c and σ are the center and width of each basis function, respectively. Further, the phase variable evolves according to

$$\tau \dot{x} = -\alpha_x x \tag{3.7}$$

where α_x is a positive constant. It can be seen that x converges exponentially to 0. Because x is a factor of $f(x)$, and the remaining factors are bounded, $f(x)$ has significant magnitude only in a finite time window. Because $f(x)$ in practice vanishes after some time, convergence to g can be concluded intuitively based on the stability of (3.2). Global exponential stability for DMPs has been shown in [Perk and Slotine, 2006; Wensing and Slotine, 2017], based on contraction theory [Lohmiller and Slotine, 1998].

Next, we will see how $f(x)$ can be determined based on a demonstrated movement. Again, this can be done separately for each dimension of the configuration, and to keep the notation uncluttered we consider a one-dimensional case. Given a discrete-time trajectory, y_demo, a corresponding DMP can be determined. Then, g is given by the end position of y_demo, whereas τ can be set to get a desired time scale. Using (3.1), the corresponding virtual force can be determined as

$$f_\mathrm{target} = \tau^2 \ddot{y}_\mathrm{demo} - \alpha(\beta(g - y_\mathrm{demo}) - \tau \dot{y}_\mathrm{demo}) = \begin{pmatrix} f^1_\mathrm{target} \\ f^2_\mathrm{target} \\ \vdots \\ f^{N_\mathrm{s}}_\mathrm{target} \end{pmatrix} \tag{3.8}$$

where N_s is the number of samples. The weights can be found by locally weighted linear regression [Atkeson et al., 1997; Schaal and Atkeson, 1998] with the solution

$$w_j = \frac{v^\mathrm{T} \Gamma_j f_\mathrm{target}}{v^\mathrm{T} \Gamma_j v} \tag{3.9}$$

where

$$v = \begin{pmatrix} x^1(g - y^1_\mathrm{demo}) \\ x^2(g - y^1_\mathrm{demo}) \\ \vdots \\ x^{N_\mathrm{s}}(g - y^1_\mathrm{demo}) \end{pmatrix} \tag{3.10a}$$

$$\Gamma_j = \mathrm{diag}(\Psi^1_j, \Psi^2_j \cdots \Psi^{N_\mathrm{s}}_j) \tag{3.10b}$$

3.1 Related Research

Thanks to its support for movement adjustment, the DMP framework has been widely used and developed within robot research, though it has yet to be established in industry. It was used for robot learning from unstructured demonstrations in [Niekum et al., 2015], where pick-and-place tasks and assembly tasks were considered. Demonstrated movements were generalized to new but static goals. It was used to reach moving goals in the context of object handover between human and robot in [Prada et al., 2014]. In [Kulvicius et al., 2012], the DMP framework was modified and a method for joining movements for tasks such as handwriting was developed. Trajectory-based reinforcement learning has been used to tune DMP parameters automatically in, *e.g.*, [Kober et al., 2008; Kroemer et al., 2010; Stulp and Schaal, 2011; Pastor et al., 2013]. In the context of robotic grasping, reinforcement learning has been used to learn goal parameters in [Stulp et al., 2011] and goals and weights simultaneously in [Stulp et al., 2012b]. Impedance was learned from experience in [Stulp et al., 2012a]. Online movement adaptation to reference force profiles was considered in [Abu-Dakka et al., 2015]. In [Chen et al., 2016], the framework for unsupervised learning of state-space models in [Karl et al., 2016] was applied to DMPs, to generate DMP representations in low-dimensional state spaces. It has also been explored how to use multiple demonstrations to refine a given movement [Matsubara et al., 2011], and how to segment movements using a library of DMPs [Meier et al., 2011].

These are just some examples where DMPs have been applied and extended. More results from previous research will be described throughout this thesis.

3.2 Parameter Choices

It has already been recommended to let $\beta = \alpha/4$. There is one degree of freedom left to be chosen, and it corresponds to the location of the double pole $-\alpha/(2\tau)$. A larger value of α corresponds to a double pole further away from the origin, and hence a faster control system. A larger value of τ has the opposite effect. Given a demonstrated trajectory, τ should be chosen before determining the weights w_j. If the same value of τ would be used later during DMP execution, the movement would be as fast as during the demonstration. One reasonable choice is to let $\tau = \alpha_x T_{\mathrm{demo}}/3$ while determining w_j, where T_{demo} is the duration of the demonstrated movement. This corresponds to 95 % convergence of x in (3.7) at time T_{demo}.

The number of basis functions N_{b} corresponds to movement resolution. In the implementations in this thesis, $N_{\mathrm{b}} = 50$ was chosen. It is reasonable to choose $c_{i,j}$ so that the basis functions have equal spacing in time, which corresponds to exponential spacing in x [Ijspeert et al., 2013]. Further, it is recommended to choose $\sigma_{i,j}$ so that at each time of the movement, some basis function is activated.

The constant α_x occurs in previous publications, but it does not affect the functionality as long as it is strictly positive, since it must be compensated for when determining DMP parameters based on desired movements. Therefore, α_x was set to 1 in the simulations and experiments in this thesis. Similarly, the initial value of x could be chosen freely and was set to 1.

3.3 Comparison with Alternative Movement Representations

A straightforward but naive approach would be to represent robot movements as time series of reference positions, to be sent to the internal robot controller during execution. The time scale of such a representation would be easy to adjust. However, it would be unclear how to adjust goal configuration and how to replan online, for instance to avoid obstacles.

Potential fields and splines are often brought up as two alternatives to DMPs for movement representation. Similar to DMPs, potential fields represent attractor landscapes, with convergence to a goal position and without explicit time dependence. Potential fields have been considered for robot control by many researchers, see, *e.g.*, [Khatib, 1986; Koditschek, 1987; Li and Horowitz, 1999]. Stability aspects of potential fields for force and motion control have been addressed in [Johansson et al., 2009; Johansson and Robertsson, 2009]. Design of potential fields for some obstacle avoidance scenarios has been done in, *e.g.*, [Koditschek, 1987]. Vector fields typically define the movement based on given positions, but determining the vector field given a desired behavior is not straightforward. Further, while DMPs allow for different control signals from the same position, this can not be achieved with conventional potential fields.

For imitation learning, splines have been widely used. Splines are functions that are defined piecewise by polynomials, and retain smoothness where the polynomials connect. It has been shown in, *e.g.*, [Miyamoto et al., 1996; Wada and Kawato, 2004] that demonstrated trajectories can be represented and successfully reproduced by means of splines. However, online replanning is not supported, and temporal and spatial scaling can be done only by recomputation of the spline polynomials.

4

Modification of Robot Movement Using Lead-Through Programming

4.1 Introduction

Similar to most software products, robot programs need to be updated from time to time. A change of product to be manufactured, a desire to shorten cycle time, previous programming mistakes, or a changed work-cell configuration are typical reasons for this. It might also be desirable to migrate functionality between different robot stations, or to be able to download preliminary programs or general templates put together elsewhere and subsequently make final adjustments for the intended robot cell. To make it worthwhile to reuse existing but imperfect robot programs rather than programming from the beginning, easy means of modification are necessary.

This chapter presents a framework that allows for a robot operator to adjust robot movements in an intuitive way. Given a faulty trajectory, the operator can use lead-through programming [Stolt et al., 2015a] to demonstrate a corrective one. Based on this, a modified trajectory representation is determined automatically. DMPs, described in [Ijspeert et al., 2013] and Chapter 3, are used to represent and generate the movement, though this presented framework does not require that the faulty trajectory is generated by a DMP.

In the process of program updating, it is natural that previously defined movements need modification. In the case where a starting point has been updated without further consideration, which in principle is supported by the DMP framework [Ijspeert et al., 2013], a DMP-generated trajectory would still converge to the goal configuration, given that no physical limitations prevent this, but it would necessarily go through a different path. The path modification would not be specified by the operator by default, and it would be larger for a larger difference between the original and the updated starting points. Similarly, if the complete trajectory is of interest it is not enough to modify the goal state. One way to solve the problem would be to record an entirely new trajectory, and then construct a corresponding DMP. However, this would be unnecessarily time consuming for the operator, if only some part of the trajectory had to be modified. In this chapter, we consider the unfavorable case where the last part of a robot trajectory is unsatisfactory. This is a realistic scenario, because many manipulation movements start by approaching a work object with large position tolerances, and end with physical contact and small position tolerances. Section 4.2 presents two such scenarios, in the context of robotic assembly.

A standard approach has been to use demonstrated configuration trajectories to determine preliminary DMPs. However, simply repeating these trajectories does not guarantee success. Further refinements and ability to adapt based on sensor data are typically necessary. Adaptation based on force measurements to follow force profiles in robotic assembly was considered in [Abu-Dakka et al., 2015]. Further, initial demonstrations have been followed by trajectory-based reinforcement learning for many different tasks. In [Pastor et al., 2013], a simulated legged robot was considered. A DMP that enabled the robot to make a small jump was first manually defined. After applying reinforcement learning, the robot could make a longer jump over a gap. The same principle was applied in [Kober et al., 2008], where a simulated robot arm was taught to catch a ball in a cup. In [Kroemer et al., 2010], reinforcement learning and imitation learning were combined to achieve object grasping in an unstructured environment. Compared to such refinements, the modification presented here is less time consuming and does not require engineering work. On the other hand, the previous work offers modulation based on sensor data and finer movement adjustment. Therefore, the framework presented in this chapter can be used as an intermediate step where, if necessary, a DMP can be modified to prepare for further improvement, see Fig. 4.1. This modification can be used within a wide range of tasks. In this chapter, we exemplify with peg-in-hole tasks.

In [Pastor et al., 2009; Ijspeert et al., 2013; Pastor et al., 2013], avoidance of spherical obstacles was incorporated in the DMP framework by modeling repulsive forces from the obstacles. In [Stavridis et al., 2017], this research was extended for avoidance of obstacles contained in ellipsoids. These approaches have been verified for several scenarios, but require an infrastructure for obstacle detection, as well as some coupling-term parameters to be defined. In practice they preserve convergence to the goal configuration in uncomplicated settings, but since the path to get there is modified by the obstacle avoidance, it is not guaranteed to follow any specific trajectory to the goal. This is often significant, for instance in assembly

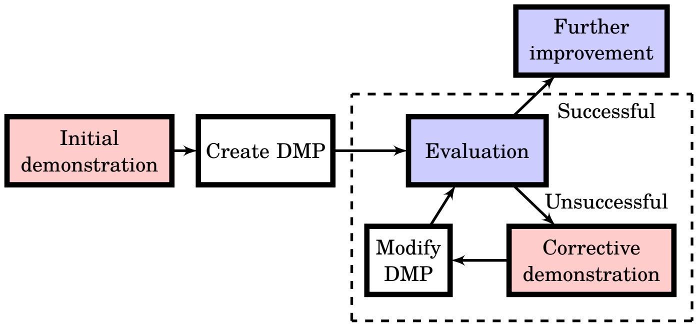


Figure 4.1 Schematic visualization of the work flow, from an operator's perspective. A DMP was created based on a demonstration. Subsequently, the DMP was executed while evaluated by the operator. If unsuccessful, the operator demonstrated a correction, which yielded a modified DMP to be evaluated. Once successful, further improvement could be done by, *e.g.*, trajectory-based reinforcement learning, though that was outside the scope of this work. Steps that required direct, continuous interaction by the operator are marked with red color. Steps that required some attention, such as supervision and initialization, are marked with blue. The operations in the white boxes were done by the software in negligible computation time, and required no human involvement. The research in this chapter focuses on the steps within the dashed rectangle.

tasks. Further, for more complex environments that are realistic in robotic manipulation, modeling repulsive forces from obstacles would imply a risk of getting stuck, thereby jeopardizing task completion. Instead, the method described here allows for the operator to lead the manipulator backwards, approximately along the part of the deficient path that should be adjusted, followed by a desired trajectory, as visualized in Fig. 4.2.

4.2 Motivating Examples

We here describe two scenarios where the framework would be useful. These also form the experimental setups considered in Secs. 4.5 and 4.6, where more details are given.

Setup 4.1 Consider the setup shown in Fig. 4.3, where a button should be placed into a yellow case. A DMP was run for this purpose, but due to any of the reasons described above the movement was not precise enough, and the robot got stuck on its way to the target. Hitherto, such a severe shortcoming would have motivated the operator to teach a completely new

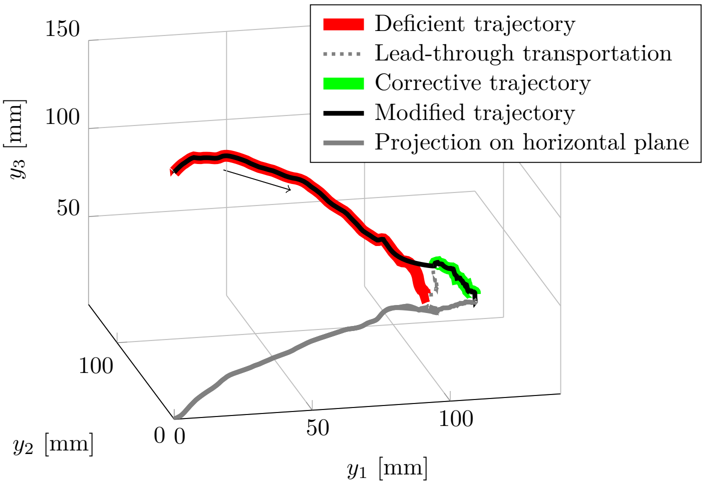


Figure 4.2 Trajectories of the robot's end-effector from one of the experiments. The arrow indicates the motion direction. A deficient trajectory was generated from an original DMP. After that, the operator demonstrated a corrective trajectory. Merging of these resulted in a modified trajectory. The projection on the horizontal plane is only to facilitate the visualization.

DMP, and erase the old one. With the method proposed in this chapter, the operator had the opportunity to approve the first part of the trajectory, and only modify the last part. This was done by leading the robot arm backwards, approximately along the faulty path, until it reached the acceptable part. Then, the operator continued to lead the arm along the desired path to the goal. When this was done, the acceptable part of the first trajectory was merged with the last part of the corrective trajectory. After that, a DMP was fitted to the resulting trajectory. Compared to just updating the target point, this approach also enabled the operator to determine the trajectory leading there.

Setup 4.2 For the setup in Fig. 4.4, there existed a DMP for moving the robot arm from the right, above the button that was already inserted, to a position just above the hole in the leftmost yellow case. However, under the evaluation the operator realized that there would have been a collision if a button were already placed in the case in the middle. A likely reason for this to happen would be that the DMP was created in a slightly different scene, where the potential obstacle was not taken into account. Further, the operator desired to extend the movement to complete the peg-in-hole task, rather than stopping above the hole. With the method described herein, the action of the operator would be similar to that in Setup 4.1, again saving work compared to previous methods.

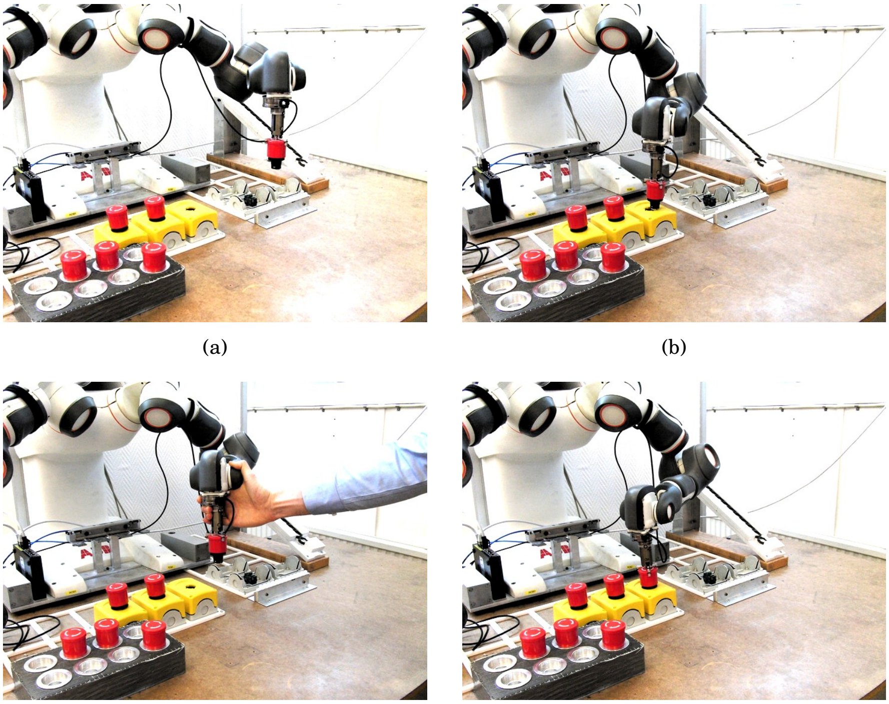
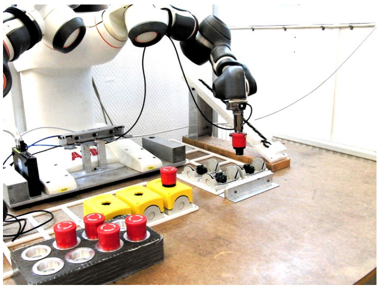
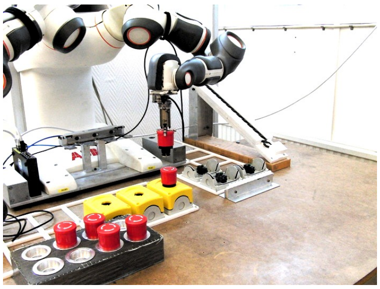


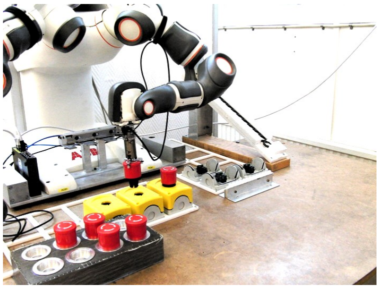
Figure 4.3 Visualization of Setup 4.1. The evaluation started in (a), and in (b) the robot failed to place the button in the hole because of inadequate accuracy. Between (a) and (b), the deficient trajectory was recorded. The operator led the robot arm backwards (c), approximately along a proportion of the deficient trajectory, and subsequently led it to place the button properly, while the corrective trajectory was recorded. The robot then made the entire motion, starting in a configuration similar to that in (a), and ending as displayed in (d). Results from this setup are shown in Figs. 4.7 and 4.8.

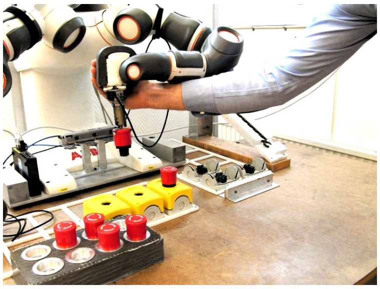
4.3 Problem Formulation

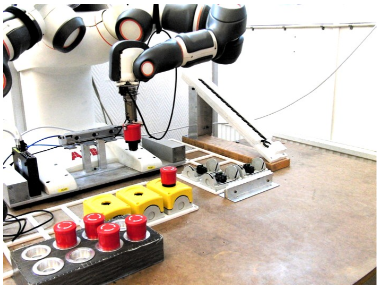
In this chapter, we address the question of whether it is possible to automatically interpret a correction, made by an operator, of the last part of a DMP trajectory, while still taking advantage of the first part. The human–robot interaction must be intuitive, and the result of a correction predictable enough for its purpose. The correction should result in a new DMP, of which the first part behaves qualitatively as the first part of the original DMP, whereas the last part resembles the corrective trajectory. Any discontinuity between the original and corrective trajectories must be mitigated.


(a)


(b)


(c)


(d)


(e)

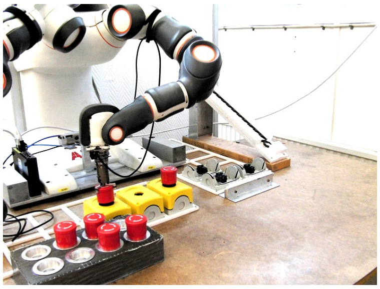

(f)

Figure 4.4 Visualization of Setup 4.2. The initial goal was to move the button to the leftmost yellow case, above the hole, to prepare for placement. The evaluation started in (a), and in (b) the trajectory was satisfactory as the placed button was avoided. In (c), however, there would have been a collision if there were a button placed in the middle case. Further, it was desired to complete the peg-in-hole task, rather than stopping above the hole. Hence, the evaluated trajectory was considered deficient. In (d), the operator led the robot arm back, and then in a motion above the potential obstacle, and into the hole, forming the corrective trajectory. Based on the modified DMP, the robot started in a position similar to that in (a), avoided the potential obstacle in (e) and reached the new target in (f). The trajectories from one attempt are shown in Fig. 4.9.

Table 4.1 Notation used in this chapter.

Notation		Description
n	$\in \mathbb{Z}^+$	Dimension of robot configuration
y	$\in \mathbb{R}^n$	Robot configuration
y_d	$\in \mathbb{R}^n$	Deficient trajectory
y_c	$\in \mathbb{R}^n$	Corrective trajectory
y_dr	$\in \mathbb{R}^n$	Retained part of deficient trajectory
y_m	$\in \mathbb{R}^n$	Modified version of y_dr
y^k	$\in \mathbb{R}^n$	Robot configuration at time sample k
τ	$\in \mathbb{R}^+$	Time parameter
g	$\in \mathbb{R}^n$	Goal state
N_b	$\in \mathbb{Z}^+$	Number of basis functions
w_j	$\in \mathbb{R}^n$	The j:th weight vector

4.4 Method

In this section a method to determine which part of a deficient trajectory to retain is presented, followed by a description of how it should be merged with a corrective trajectory to avoid discontinuities. Finally, some implementation aspects are addressed. Figure 4.1 displays a schematic overview of the work flow of the application, from the user's perspective. Table 4.1 lists some of the notation used in this chapter.

Interpretation of Corrective Demonstration

Consider the case where a deficient trajectory, denoted y_d, has been generated. The operator would then use lead-through [Stolt et al., 2015a] to move the robot arm backwards, approximately along the deficient path, until it reached an acceptable part. We refer to this movement as lead-through transportation, as opposed to lead-through programming, because the purpose of this movement is not to demonstrate directly but rather to prepare for demonstration. Hence, it is not necessary to log the resulting trajectory, though it was done in the experiments for the purpose of visualization. Thereafter, lead-through programming was used to demonstrate a desired path to the goal. This movement is referred to as the corrective trajectory, denoted y_c. Examples of these trajectories are shown in Figs. 4.2, 4.5 and 4.6. A trajectory formed by simply appending y_c to y_d would likely take an unnecessary detour. Thus, only the first part of y_d was retained. This is illustrated in Fig. 4.6. The retained part, denoted y_dr, was determined as the part previous to the sample of y_d that was closest to the initial point of y_c, denoted y_c^1, *i.e.*,

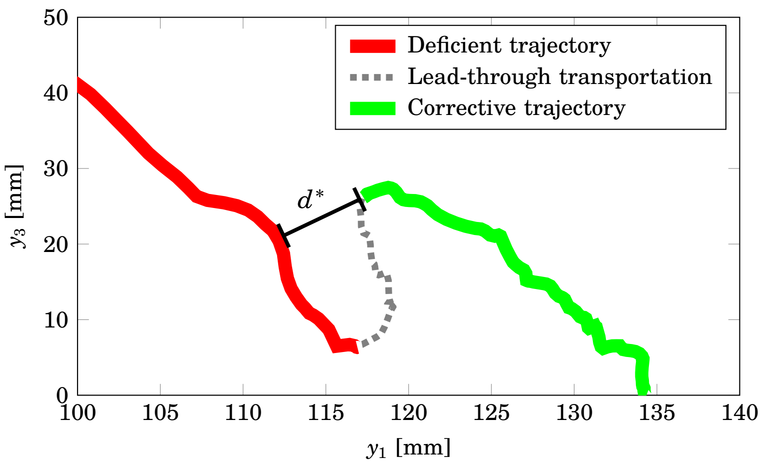


Figure 4.5 Visualization of shortest distance, here denoted d^*, used to determine the left separation marker in Fig. 4.6. The trajectories are the same as in Figs. 4.2 and 4.6, except that the modified trajectory is omitted here.

$$y_{\mathrm{dr}}^m = y_{\mathrm{d}}^m, \quad \forall m \in \{1, 2 \ldots M\} \tag{4.1}$$

where

$$M = \underset{k=1,2\ldots K}{\operatorname{argmin}} \; d(y_{\mathrm{d}}^k, y_{\mathrm{c}}^1) \tag{4.2}$$

Here, $d(\cdot,\cdot)$ denotes distance between two configurations, and K is the number of samples in y_{d}; see Fig. 4.5 for an illustration. The approach of using the shortest distance as a criterion, was motivated by the assumption that the operator led the robot arm back, approximately along the deficient trajectory, until the part that was satisfactory. At this point, the operator started the corrective demonstration, thus defining y_{c}^1 (see right marker in Fig. 4.6). By removing parts of the demonstrated trajectories, a significant discontinuity between the remaining parts was introduced. In order to counteract this, y_{dr} was modified into y_{m}, of which the following three features were desired:

1. y_{m} should follow y_{dr} approximately;
2. The curvature of y_{m} should be moderate;
3. y_{m} should end where y_{c} began, with the same movement direction in this point.

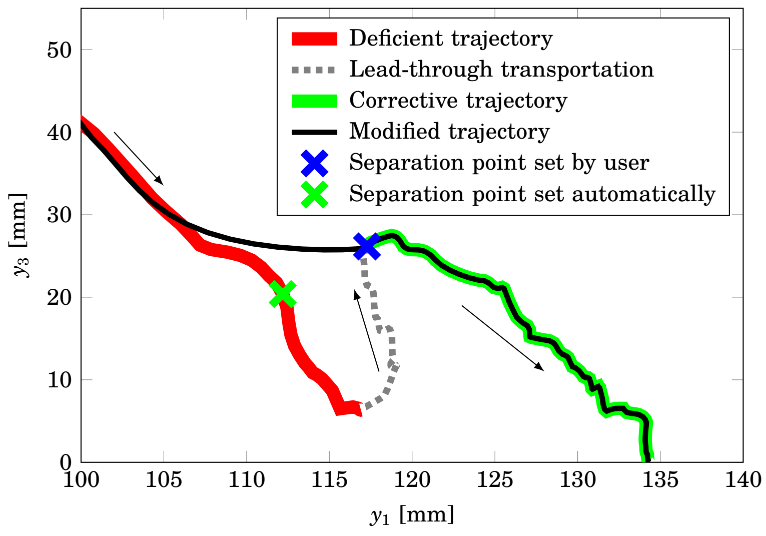


Figure 4.6 Same trajectories as in Fig. 4.2, but zoomed in on the corrective trajectory. Arrows indicate directions. The parts of the trajectories between the separation markers were not retained. The right, blue, separation point was determined explicitly by the operator during the corrective demonstration. The left, green, separation point was determined according to (4.2). Further, what was left of the deficient trajectory was modified for a smooth transition. However, the part of the corrective trajectory retained was not modified, since it was desired to closely follow this part of the demonstration. Note that the trajectories retained were not intended for direct play-back execution. Instead, they were used to form a modified DMP, which in turn generated a resulting trajectory, as shown in Figs. 4.7 to 4.9.

To find a suitable trade-off between these objectives, the following convex optimization problem was formulated and subsequently solved, for each dimension of the configuration:

$$\underset{y_\mathrm{m}}{\text{minimize}} \quad \|y_\mathrm{dr} - y_\mathrm{m}\|_2 + \lambda \|T_{(\Delta^2)} y_\mathrm{m}\|_2 \tag{4.3a}$$

$$\text{subject to} \quad y_\mathrm{m}^M = y_\mathrm{c}^1 \tag{4.3b}$$

$$y_\mathrm{m}^M - y_\mathrm{m}^{M-1} = y_\mathrm{c}^2 - y_\mathrm{c}^1 \tag{4.3c}$$

Here, $T_{(\Delta^2)}$ is a second-order finite-difference operator, and λ denotes a constant scalar. A larger value of λ corresponds to higher priority of Feature 2 as compared to Feature 1. Thereafter, y_c was appended to y_m, and one corresponding DMP was created with the method described in [Ijspeert et al., 2013] and Chapter 3. The next step in the work flow was to evaluate the resulting DMP, as shown in Fig. 4.1.

Software Implementation

Most of the programming was done in C++, where DMPs were stored as objects of a class. Among the data members of this class were the parameters τ, g, and $w_{1 \ldots N_\mathrm{b}}$, as well as some description of the context of the DMP and when it had been created. It contained member functions for displaying the parameters, and for modifying g and τ. The C++ linear algebra library Armadillo [Sanderson and Curtin, 2016] was used in a major part of the implementation. Further, the code generator CVXGEN [Mattingley and Boyd, 2012] was used to generate C code for solving the optimization problem in (4.3). By default, the solver code was optimized with respect to computation time. This resulted in a real-time application, in which the computation times were negligible in teaching scenarios. The optimization problem was typically solved well below one millisecond on an ordinary PC. The implementation developed in [Stolt et al., 2015a] was used to enable lead-through programming. The operator could switch between different high-level modes such as DMP execution, lead-through transportation, and corrective movement by pressing buttons in a terminal user interface. During lead-through programming, trajectory logging started when a velocity significantly different from zero was achieved.

4.5 Experiments

The robot used in the experimental setup was a prototype of the dual-arm ABB YuMi [ABB Robotics, 2018], described in Chapter 2. The computations took place in joint space, and the robot's forward kinematics were used for visualization in Cartesian space in the figures presented in this chapter. The setups in Sec. 4.2 were used to evaluate the proposed method. For each trial, the following steps were taken:

- An initial trajectory was taught, deliberately failing to meet the requirements, as explained in Sec. 4.2;
- Based on this, a DMP was created;
- The DMP was used to generate a trajectory similar to the initial one. This formed the deficient trajectory;
- The arm was retracted by means of lead-through transportation;
- A corrective trajectory was recorded;

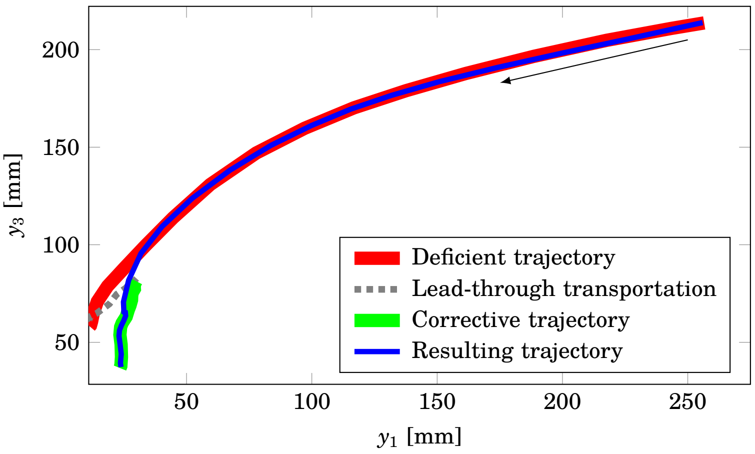


Figure 4.7 Trajectories from the experimental evaluation of Setup 4.1. The deficient trajectory went past the goal in the negative y_1-direction, preventing the robot from lowering the button into the hole. After correction, the robot was able to reach the target as the modified DMP generated the resulting trajectory.

- Based on the correction, a resulting DMP was formed automatically;
- The resulting DMP was executed for experimental evaluation.

First, Setup 4.1 was set up for evaluation, see Fig. 4.3. The scenario started with execution of a deficient trajectory. For each attempt, a new deficient trajectory was created and modified. A total of 50 attempts were made.

Similarly, Setup 4.2 (see Fig. 4.4) was set up, and again, a total of 50 attempts were made.

A video is available as a publication attachment to [Karlsson et al., 2017b], to facilitate understanding of the experimental setup and results. A version with higher resolution is available on [Karlsson, 2017d].

4.6 Results

For each attempt of Setup 4.1, the robot was able to place the button properly in the yellow case after the modification. Results from two of these attempts are shown in Figs. 4.7 and 4.8. In the first case, the deficient trajectory went past the goal, whereas in the second case, it did not reach far enough.

Each of the attempts of Setup 4.2 was also successful. After modification, the DMPs generated trajectories that moved the grasped stop button

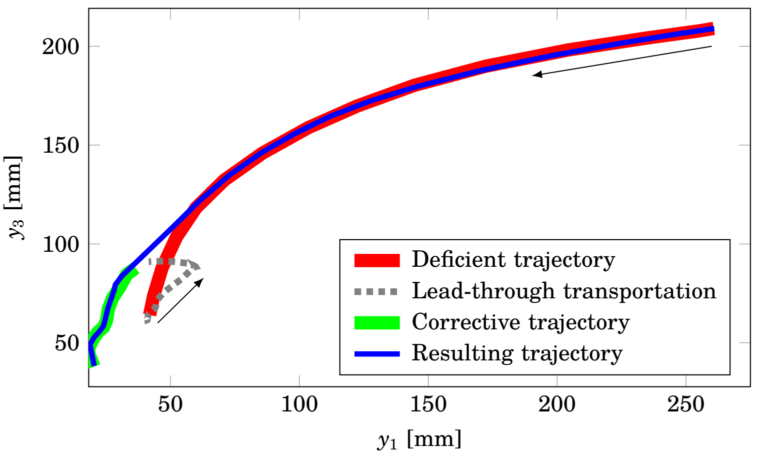


Figure 4.8 Similar to Fig. 4.7, except that in this case, the deficient trajectory did not reach far enough in the negative y_1-direction.

above the height of potential obstacles, in this case other stop buttons, and subsequently inserted it into the case. The result from one attempt is shown in Fig. 4.9.

4.7 Discussion

Future work includes integration of the presented framework with reinforcement learning [Kober et al., 2008; Kroemer et al., 2010; Pastor et al., 2013], in order to optimize the motion locally with respect to criteria such as execution time. The program should also be augmented to take the purpose of, and relation between, different DMPs into consideration. This extension will emphasize the necessity of keeping track of different states within the work flow. To this purpose, a state machine implemented in, *e.g.*, JGrafchart [Theorin, 2014], or the framework of behavior trees, applied on an aerial vehicle in [Ögren, 2012] and robot control in [Marzinotto et al., 2014], would be suitable.

Similar to movement adjustments based on sensor data, adjustments by the operator during DMP execution could also be considered. Such a step was taken in [Talignani Landi et al., 2018a], where [Karlsson et al., 2017b] was extended significantly to enable adjustment by physical contact during movement. The software in Sec. 4.4, first implemented in [Karlsson et al., 2017b], was also used as a starting point for the implementation in [Talignani Landi et al., 2018a]. A presentation of [Talignani Landi et al., 2018a] in video format, including a demonstration of the functionality, is

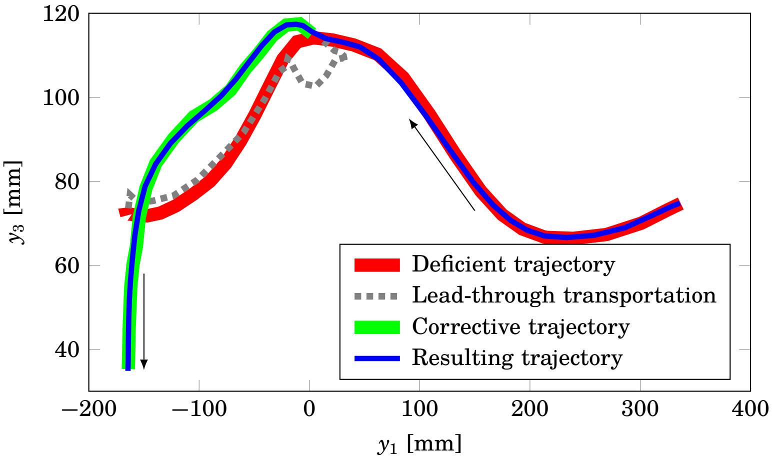


Figure 4.9 Trajectories from experimental evaluation of Setup 4.2. For the deficient trajectory, the end-effector was lowered too early, causing a potential collision. After the correction, the robot was able to reach the target while avoiding the obstacles. The movement was also extended to perform the entire peg-in-hole task, rather than stopping above the hole.

available in [Talignani Landi et al., 2018b].

When robotic movement is adjusted in real time, whether it is due to exploration in reinforcement learning, physical interaction with a human, or something else, it is problematic that the phase variable x evolves independently of this for the ordinary DMP formulation, see [Ijspeert et al., 2013]. This issue will be addressed in Chapter 5.

Performing the computations in joint space instead of Cartesian space allowed for the operator to determine the entire configuration of the seven-degree-of-freedom robot arm, rather than the pose of the tool only. However, one could think of situations where the operator is not concerned about the configuration, and the pose of the tool would be more intuitive to consider. It would therefore be valuable if it could be determined whether the operator aimed to adjust the configuration or just the pose of the tool. For example, a large configuration change yielding a small movement of the tool, should promote the hypothesis that the operator aimed to adjust the configuration.

It should be stated that the scenarios evaluated here are not covering the whole range of plausible scenarios related to this method, and it remains as future work to investigate the generalizability, and user experience, more thoroughly. The last part of the resulting movement is guaranteed to follow the retained part of the corrective demonstration accurately, given enough DMP basis functions. Hence, the only source of error on that part is a faulty demonstration. For instance, the movement might require higher accuracy than what is possible to demonstrate using lead-through programming. Another limitation with this method is that it is difficult for the operator to very accurately determine which part of the faulty trajectory to retain, since this is done autonomously. However, for the experiments performed here, the estimation by the operator was sufficient to demonstrate the desired behavior. The benefit with this approach is that it saves time as the operator does not have to specify all details explicitly.

4.8 Conclusion

In this chapter, an approach for modification of DMPs, using lead-through programming, was presented. It allowed for a robot operator to modify faulty trajectories, instead of demonstrating new ones from the beginning. Based on corrective demonstrations, modified DMPs were formed automatically. A real-time application, that did not require any additional engineering work by the user, was developed and verified experimentally. A video showing the functionality is available online [Karlsson, 2017d].

5

Temporally Coupled Dynamical Movement Primitives in $\mathbb{R}^n$

5.1 Introduction

In this chapter, we continue to extend the DMP framework to support movement adjustment. Whereas Chapter 4 focused on corrective demonstrations after faulty movements, this chapter aims to allow for replanning during movement.

Consider the case where a robot has to deviate from its intended movement. One could think of several motivations for such deviation, for instance avoiding an obstacle, avoiding blocking camera vision, waiting for a work object to be handed over from another robot, *etc*. In this thesis, we use physical contact with a human to exemplify. A human grasps or pushes the robot, thereby stopping its movement or pushing it away from its path.

Typical industrial robots would react to this in two possible ways. A large robot could possibly continue its movement despite the physical contact, because of its large mass and strength compared to the human. A lightweight collaborative robot, which is mainly considered in this thesis, has weaker motors and its movement would be significantly affected by the human. The motors would not be counteracted by the human for long, however, because the robot would typically stop if reference states were too far from measured states, or if joint torques were unexpectedly high. The robot motors would then deactivate, the brakes would activate, and the robot would possibly provide the operator with an error message. This is a safety feature, sometimes called motion supervision, with the purpose of avoiding damage of the robot and its surroundings. Motion supervision runs on larger robots as well, though these are less sensitive to external forces. Triggering such an error in an industrial setting implicates production stop and significant engineering work to reset and adjust hardware and software. Hence, this strategy is not feasible in less predictable environments.

Already the original DMP formulation implicates some replanning capabilities, see [Ijspeert et al., 2013] and Chapter 3. It is possible to update the goal configuration during movement as utilized in [Prada et al., 2014], and to avoid uncomplicated obstacles [Pastor et al., 2009; Ijspeert et al., 2013; Pastor et al., 2013]. DMPs with compliant behavior were developed in [Denisa et al., 2016; Batinica et al., 2017]. Further, movement can be generated from any configuration with guaranteed exponential convergence to the goal [Perk and Slotine, 2006; Ijspeert et al., 2013; Wensing and Slotine, 2017]. Therefore, it was possible to physically push a robot to correct its movement in [Talignani Landi et al., 2018a], knowing that it would reach the goal after the perturbation.

However, in the original formulation, a DMP would continue its time evolution despite any significant perturbations. The behavior of the robot would then likely be undesirable and not intuitive, as discussed in [Ijspeert et al., 2013]. A solution was also proposed in [Ijspeert et al., 2013]. Intuitively, the time evolution of the DMP was slowed down in case of deviation from the intended trajectory. This is called temporal coupling, of which phase stopping is one component. In this chapter, it is discovered that the temporal coupling proposed in [Ijspeert et al., 2013] needs to be extended to be realizable in practice, though the core concept in [Ijspeert et al., 2013] is very promising and therefore forms a foundation for the method developed in this chapter.

This chapter provides a practically realizable extension of the temporal coupling in [Ijspeert et al., 2013], in the form of a two-degree-of-freedom motion control system; see [Åström and Wittenmark, 2013] for a description of the general architecture for two-degree-of-freedom control. The feedforward part of the controller promotes tracking of the DMP trajectory in the absence of significant perturbations, thus mitigating unnecessarily slow trajectory evolution due to temporal coupling acting on small tracking errors. The feedback part suppresses significant errors. It is further shown theoretically that the control system is globally exponentially stable, which in practice means that the goal configuration is guaranteed to be reached.

This chapter is organized as follows. Previous research is described and evaluated in Sec. 5.2. The research problem addressed in this chapter is specified in Sec. 5.3. The proposed control algorithm is presented in Sec. 5.4. Thereafter, the proposed control algorithm is compared with previous research through simulations in Sec. 5.5. These simulations also support understanding of the key concepts of temporally coupled DMPs, and this is why they are presented before the mathematical stability analysis in Sec. 5.6. Further, experiments and results are presented in Secs. 5.7 and 5.8, a discussion is presented in Sec. 5.9, and concluding remarks are

Table 5.1 Notation used in this chapter.

Notation		Description
n	$\in \mathbb{Z}^+$	Dimension of robot configuration
y	$\in \mathbb{R}^n$	Physical robot configuration
$\ddot{y}_\mathrm{r}$	$\in \mathbb{R}^n$	Robot acceleration reference
τ	$\in \mathbb{R}^+$	Time parameter
g	$\in \mathbb{R}^n$	Goal state
x	$\in \mathbb{R}^+$	Phase variable
$f(x)$	$\in \mathbb{R}^n$	Learnable virtual forcing term
α, β, α_x	$\in \mathbb{R}^+$	Positive constants
e	$\in \mathbb{R}^n$	Low-pass filtered configuration error
y_c	$\in \mathbb{R}^n$	Coupled robot configuration
τ_a	$\in \mathbb{R}^+$	Adaptive time parameter
$\alpha_e, k_t, k_\mathrm{c}$	$\in \mathbb{R}^+$	Positive constants
z	$\in \mathbb{R}^n$	DMP state
$K_\mathrm{p}, K_\mathrm{v}$	$\in \mathbb{R}^+$	Large control gains
$k_\mathrm{p}, k_\mathrm{v}$	$\in \mathbb{R}^+$	Moderate control gains
y_u	$\in \mathbb{R}^n$	Unperturbed, uncoupled trajectory
N_b	$\in \mathbb{Z}^+$	Number of basis functions
ξ	$\in \mathbb{R}^{5n+1}$	DMP state vector
$\lVert\xi_i\rVert$	$\in \mathbb{R}^+$	2-norm of DMP state i

finally presented in Sec. 5.10. Table 5.1 lists some of the notation used in this chapter.

Successful perturbation recovery using the proposed method is visualized in a video available online [Karlsson, 2016]. The video will be explained in more detail in this chapter, but could already at this point support understanding of the functionality.

A code example is available in [Karlsson, 2017c], to allow for exploration of the system proposed. The system was also integrated in the Julia DMP package in [Bagge Carlson, 2016], originally based on [Ijspeert et al., 2013].

5.2 Previous Research on Temporally Coupled DMPs for Perturbation Recovery

We here consider the case where a perturbation is introduced, so that the reference acceleration $\ddot{y}_\mathrm{r}$ in

$$\tau^2 \ddot{y}_\mathrm{r} = \alpha(\beta(g - y) - \tau \dot{y}) + f(x) \tag{5.1}$$

can not be realized, *i.e.*, the robot configuration y can not evolve according to plan. In the original DMP formulation, the phase variable would evolve independently of any perturbation, according to

$$\tau \dot{x} = -\alpha_x x \tag{5.2}$$

This causes undesired behavior, since it is then likely that the actual trajectory y deviates significantly from the intended trajectory even after the cause of the perturbation has vanished. This is more thoroughly described in [Ijspeert et al., 2002; Ijspeert et al., 2013]. Consider, for instance, the case where the robot is stopped during its movement until x has converged to 0. This implies that $f(x) = 0$, and once released the robot will move to its goal without taking the parameters of $f(x)$ into account. In such a case, the value of the phase variable does not correspond to the progress of the movement. To mitigate this problem, the solution described in the following paragraph was suggested in [Ijspeert et al., 2013].

A coupling term C_t and an adaptive time parameter τ_a were introduced.

$$\dot{e} = \alpha_e(y - y_\mathrm{c} - e) \tag{5.3a}$$

$$C_t = k_t e \tag{5.3b}$$

$$\tau_\mathrm{a} = 1 + k_\mathrm{c} e^\mathrm{T} e \tag{5.3c}$$

Here, α_e, k_t, and k_c are constant parameters, and e is low-pass filtered position error. The parameter τ_a is used to determine the evolution rate of the entire dynamical system. Further, y_c denotes a coupled version of y, and it is a simulated trajectory that can adjust its evolution rate based on τ_a. In contrast, y continues to represent the physical trajectory of the robot. Further, the evolution of x was determined by

$$\tau_\mathrm{a} \dot{x} = -\alpha_x x \tag{5.4}$$

A larger value of e yields a larger value of τ_a and hence slower evolution rate. One important aspect is that τ_a and x are scalars, that are shared across all degrees of freedom of the movement. This supports coordination between the different degrees of freedom. Note that (5.1) can be written as a first-order dynamical system, given that the reference acceleration is realized.

$$\tau \dot{z} = \alpha(\beta(g - y) - z) + f(x) \tag{5.5a}$$

$$\tau \dot{y} = z \tag{5.5b}$$

Based on this, the dynamics of y_c were defined as follows.

$$\tau_\mathrm{a} \dot{z} = \alpha(\beta(g - y_\mathrm{c}) - z) + f(x) + C_t \tag{5.6a}$$

$$\tau_\mathrm{a} \dot{y}_\mathrm{c} = z \tag{5.6b}$$

A PD controller given by

$$\ddot{y}_\mathrm{r} = K_\mathrm{p}(y_\mathrm{c} - y) + K_\mathrm{v}(\dot{y}_\mathrm{c} - \dot{y}) \tag{5.7}$$

was used to drive y to y_c. Here, $\ddot{y}_\mathrm{r}$ denotes reference acceleration, while K_p and K_v are control gains.

The ability to slow down the evolution of x in (5.4) is sometimes called phase stopping [Abu-Dakka et al., 2015], whereas the entire functionality of affecting the DMP evolution rate through τ_a and retaining coordination between the degrees of freedom is called temporal coupling [Ijspeert et al., 2013].

Evaluation of Previous Research

The approach from previous research has taken several important parts of perturbation recovery into account, and it should be emphasized that it forms the foundation of the research presented in this chapter. In this section, however, some aspects are considered that motivate extension of the state-of-the-art framework.

Denote by y_u an unperturbed trajectory generated by an uncoupled DMP, according to (5.1). In the absence of significant perturbations, it is desirable that the trajectory generated by a temporally coupled DMP, y, evolves in the same way as y_u. To achieve this, y should follow y_c closely. If this would not be achieved, in addition to the deviation itself, y and y_c would be slowed down compared to y_u, due to the temporal coupling according to (5.3c). This phenomenon is shown in Sec. 5.5. In [Ijspeert et al., 2013], very high control gains for (5.7) were suggested, which would have mitigated this issue under ideal conditions and unlimited magnitude of the control signals. Specifically, $K_\mathrm{p} = 1000$ and $K_\mathrm{v} = 125$ were chosen. However, even for moderate perturbations, this would imply control signals too large to be realized practically. For instance, a position error in Cartesian space of 0.1 m would yield $\ddot{y}_\mathrm{r} = 100\,\mathrm{m/s^2}$, whereas the YuMi robot has a maximum end-effector acceleration in the order of $10\,\mathrm{m/s^2}$. In Figs. 5.1 and 5.2, two simulated examples are displayed: one where the actual movement was stopped, and one where it was moved away from the intended path. The method described in [Ijspeert et al., 2013] was used for recovery, and prohibitively large values of $\ddot{y}_\mathrm{r}$ were generated. This control system is also sensitive to noise and has a dangerously low delay margin of 12 ms. The simulations are more thoroughly described in Sec. 5.5.

5.3 Problem Formulation

In this chapter, we address the question of whether perturbations of DMPs could be recovered from, while fulfilling the following requirements. Only

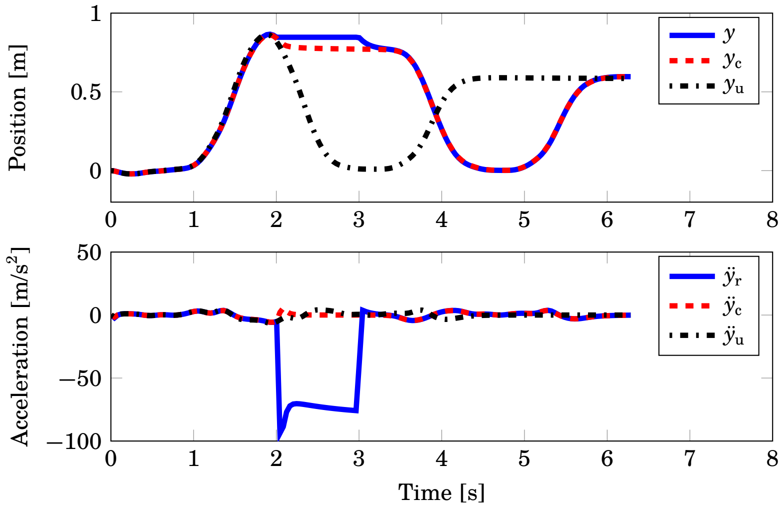


Figure 5.1 Simulated trajectories, where y was subjected to a stopping perturbation from 2 s to 3 s, using the approach in [Ijspeert et al., 2013]. When y was stopped, the evolution of y_c slowed down, and when y was released, it was driven to y_c and then behaved like a delayed version of y_u. This behavior was desired. However, a prohibitively large acceleration reference $\ddot{y}_r$ was generated.

moderate control signals must be used. The benefits of the DMP framework described in [Ijspeert et al., 2013], *i.e.*, scalability in time and space as well as guaranteed convergence to the goal g, must be preserved. Further, in the absence of significant perturbations, the behavior of y should resemble that of the original DMP framework described in [Ijspeert et al., 2013] and Chapter 3.

The stability of the proposed control algorithm requires special attention. Unstable robot motion control could damage the robot and its surroundings, such as tooling and work pieces. Further, in robotic manipulation it is crucial that the robot reaches its goal configuration in each of its movements. If this would not be achieved, subtasks would likely be left incomplete, yielding unforeseen hardware configurations, which in turn could result in collision and broken hardware. For temporally coupled DMPs to be used on a larger scale in the future, it is therefore necessary to prove that these result in stable behavior. Stability for the original DMP framework, *i.e.*, not including temporal coupling, was concluded in [Perk and Slotine, 2006; Wensing and Slotine, 2017].

Specifically, it will in this chapter be investigated whether the proposed temporally coupled DMPs are globally exponentially stable. A mathematical

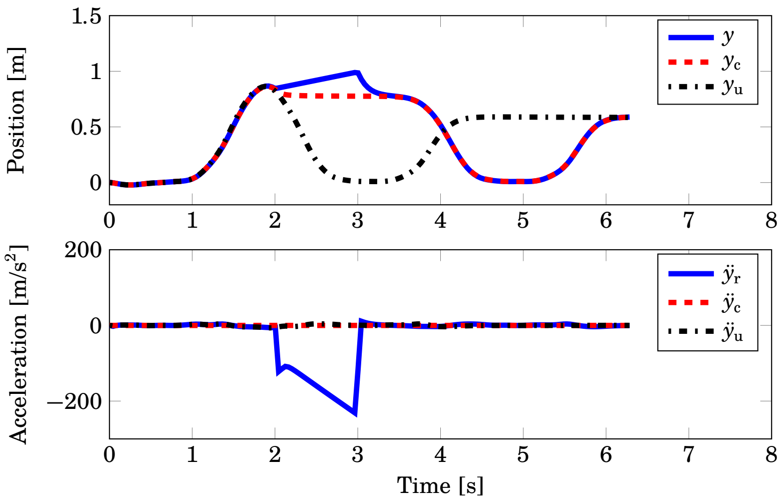


Figure 5.2 Similar to Fig. 5.1, except that y was moved away from the intended path between 2 s and 3 s. Again, a prohibitively large acceleration reference $\ddot{y}_r$ was generated.

definition of exponential stability can be found in, *e.g.*, [Slotine and Li, 1991]. In words, a system is globally exponentially stable if the state vector converges to the origin faster than an exponentially decaying function. In the context of DMPs, global exponential stability implicates that the robot eventually reaches the goal configuration.

5.4 Method

The proposed method extends the approach in [Ijspeert et al., 2013] as follows. The PD controller in (5.7) is augmented with feedforward control, as shown in (5.8). Further, the PD controller gains are moderate, to get a practically realizable control signal. Additionally, the time constant τ is introduced as a factor in the expression for the adaptive time parameter τ_a, see (5.3c) and (5.9). Our method is detailed in the following.

To enforce y to follow y_c, we apply the following control law.

$$\ddot{y}_r = k_p(y_c - y) + k_v(\dot{y}_c - \dot{y}) + \ddot{y}_c \tag{5.8}$$

Here, $\ddot{y}_c$ is obtained from feedforward of the acceleration of y_c. This allows for the controller to act also for zero position and velocity errors. Therefore, the trajectory tracking works also for moderate controller gains: $k_p = 25$ and $k_v = 10$ are used throughout this chapter. With these gains, the closed

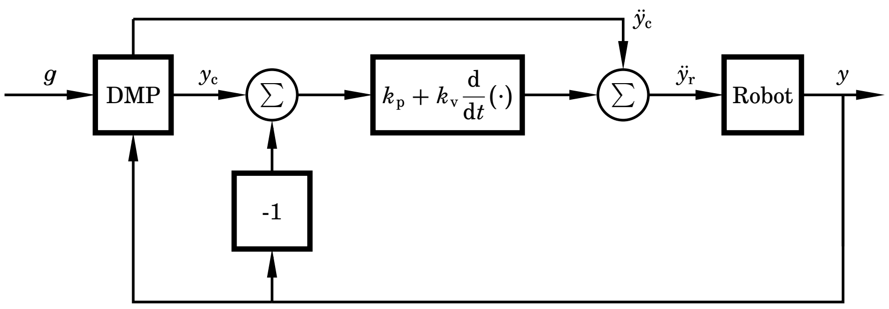


Figure 5.3 Schematic overview of the control structure described in Sec. 5.4. The block denoted 'Robot' includes the internal controller of the robot.

control loop has a double pole given by −5 rad/s. Since the real parts are negative, the system (5.8) is asymptotically stable, and since the imaginary parts are 0, it is critically damped. Such a critically damped system is obtained for $k_p = k_v^2/4$. This relationship is advisable to fulfill in the implementation, since overshoot is then avoided, which is important for avoiding collision. This is especially important in manipulation that involves contact between robot and environment; see also Chapter 3, where α and β are constrained based on the same reasoning. The delay margin is 130 ms, which is an improvement compared to 12 ms for the previous method, described in Sec. 5.2. A schematic overview of the control system is shown in Fig. 5.3.

Further, the relation (5.3c) is modified in order to include the original time constant τ, as follows.

$$\tau_a = \tau(1 + k_c e^T e) \tag{5.9}$$

The coupling term C_t is omitted in this present method. This choice is elaborated on in Sec. 5.9.

Since τ_a is not constant over time, determining $\ddot{y}_c$ is more complicated than determining $\ddot{y}$ by differentiating (5.5b). One option would be to approximate $\ddot{y}_c$ by discrete-time differentiation of $\dot{y}_c$. However, instead we determine the instantaneous acceleration analytically as follows.

$$\ddot{y}_c = \frac{d}{dt}(\dot{y}_c) = \frac{d}{dt}\left(\frac{z}{\tau_a}\right) = \frac{\tau_a \dot{z} - \dot{\tau}_a z}{\tau_a^2} = \frac{\dot{z}\tau_a - 2\tau k_c (e^T \dot{e}) z}{\tau_a^2} \tag{5.10}$$

where $\dot{e}$ and $\dot{z}$ are given by (5.3a) and (5.6a) with $C_t = 0$, respectively. It is noteworthy that the computation of $\ddot{y}_c$ does not require any first or second-order time derivatives of any measured signal, which would have required prior filtering to mitigate amplification of high-frequency noise. Similarly, $\dot{y}_\mathrm{c}$ is determined by (5.6). In contrast, the computation of $\dot{y}$ should be complemented with a low-pass filter, to mitigate amplification of measurement noise.

We model the 'Robot' block in Fig. 5.3 as a double integrator, so that $\ddot{y} = \ddot{y}_\mathrm{r}$, see Sec. 5.9. In summary, the proposed control system is given by

$$\ddot{y}_\mathrm{r} = k_\mathrm{p}(y_\mathrm{c} - y) + k_\mathrm{v}(\dot{y}_\mathrm{c} - \dot{y}) + \ddot{y}_\mathrm{c} \tag{5.11a}$$

$$\dot{e} = \alpha_e(y - y_\mathrm{c} - e) \tag{5.11b}$$

$$\tau_\mathrm{a} = \tau(1 + k_\mathrm{c} e^\mathrm{T} e) \tag{5.11c}$$

$$\tau_\mathrm{a}\dot{x} = -\alpha_x x \tag{5.11d}$$

$$\tau_\mathrm{a}\dot{y}_\mathrm{c} = z \tag{5.11e}$$

$$\tau_\mathrm{a}\dot{z} = \alpha(\beta(g - y_\mathrm{c}) - z) + f(x) \tag{5.11f}$$

Feedforward control has been used in the DMP context previously, but then only for low-level joint control, with motor-torque commands as control signals, see [Park et al., 2008; Pastor et al., 2009]. This control structure was also applied in the internal controller used in the implementation in this chapter. This inner control design should not be confused with the feedforward control described in this section, which operated outside the internal robot controller, and was used to determine the reference acceleration for the robot.

5.5 Simulations

In the simulations, the robot configuration consisted of position in Cartesian space. First, a demonstrated trajectory, y_demo, was simulated by creating a time series of positions. Thereafter, the corresponding DMP parameters were determined. Simulating the resulting DMP without perturbations and without temporal coupling yielded an unperturbed, uncoupled, trajectory y_u. Position in one dimension was simulated in the following, and two different perturbations were considered: one where y was stopped, and one where it was moved. The perturbations took place from time 2 s to 3 s. The systems were sampled at 250 Hz, using the forward Euler method. The same DMP, yielding the same y_u, was used in each trial. The adaptive time parameter τ_a was determined according to (5.9) in all simulations, to get comparable time scales. First, the controller detailed in [Ijspeert et al., 2013] was applied. Except for the perturbations themselves, the conditions were assumed to be ideal, *i.e.*, no delay and no noise were present. The results are shown in Figs. 5.1 and 5.2. Despite ideal conditions, prohibitively large accelerations were generated by the controller in both cases.

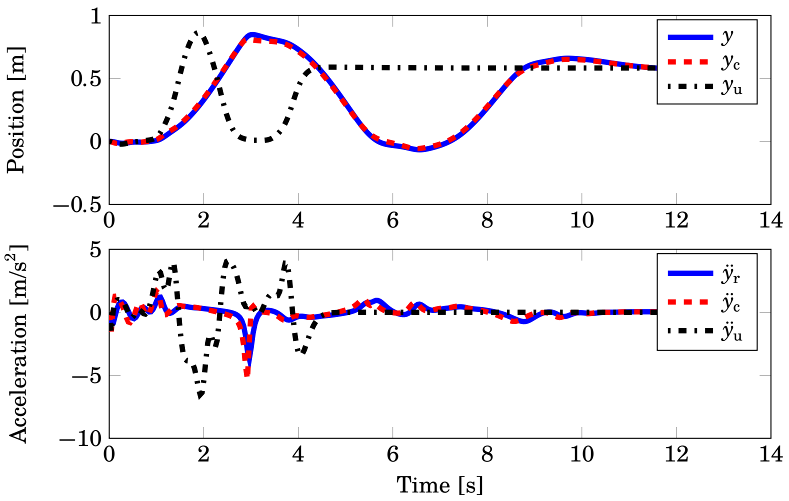


Figure 5.4 Simulation with the control structure in [Ijspeert et al., 2013], except that the gains were lower ($K_p = 25$ and $K_v = 10$). The reference acceleration was of realizable magnitude. For instance, the YuMi robot has a maximum end-effector acceleration of approximately 10 m/s^2. However, the coupled and real systems were slowed down due to small tracking errors combined with temporal coupling.

Figure 5.4 shows the result from a simulation where the controller detailed in [Ijspeert et al., 2013] was used, except that the gains were lowered to moderate values. The conditions were ideal, and no perturbation was present. This resulted in realizable control signals. However, small control errors in combination with the temporal coupling slowed down the evolution of the coupled system as well as the actual movement.

Thereafter, the controller proposed in this chapter, described in Sec. 5.4, was used. In order to verify robustness under realistic conditions, noise and time delay were introduced. Position measurement noise, and velocity process noise, were modeled as zero-mean Gaussian white noise, with standard deviations of 1 mm and 1 mm/s, respectively. The time delay between the process and the controller was 12 ms. This delay was suitable to simulate since it corresponds both to the delay margin of the method suggested in [Ijspeert et al., 2013], and to the actual delay in the implementation presented in this chapter, see Sec. 5.7. (It is, however, a coincidence that these two have the same value. Nevertheless, this shows that a 12 ms delay margin is not necessarily enough.) The results are shown in Figs. 5.5 and 5.6. For comparison, the method in [Ijspeert et al., 2013], with the large gains, was also evaluated under these conditions, although without

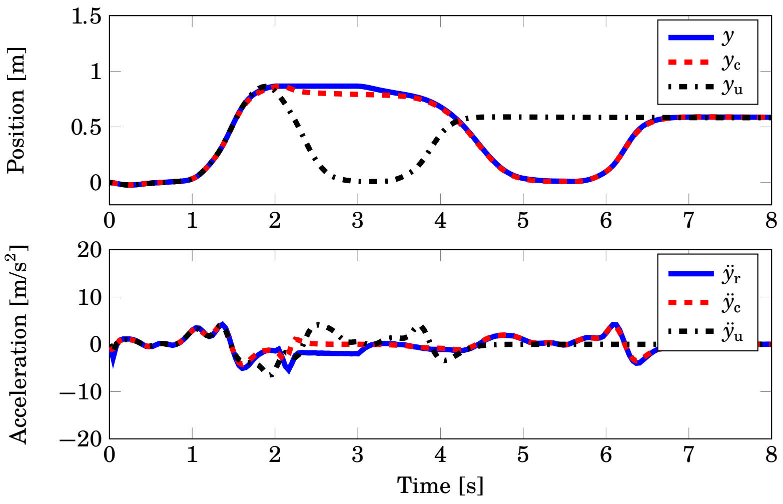


Figure 5.5 Similar to Fig. 5.1, but with modeled noise and delay, and using the controller proposed in this chapter. The behavior was satisfactory both regarding position and acceleration.

any perturbation except for the noise. Because of the time delay, this system was unstable, as shown in Fig. 5.7.

Finally, two-dimensional movement using the proposed control system was simulated. The purpose of these simulations was to investigate the convergence of the proposed control system. At the start of each simulation, y_c was set to the beginning of the demonstrated trajectory y_{demo}, and y was deliberately initialized with a randomly chosen error with respect to y_c. The system was simulated 100 times. The simulations gave mutually similar results. The results from some of the simulations are visualized in Figs. 5.8 and 5.9. For each simulation, the position converged to the goal. Further, the DMP states in Fig. 5.9 converged to 0. This is consistent with the theory that will be presented in Sec. 5.6.

5.6 The Proposed Temporally Coupled DMPs are Globally Exponentially Stable

In this section, we analyze the stability properties of the proposed temporally coupled DMPs. This analysis requires that the state space is contractible, which is true for $\mathbb{R}^n$ (see Chapter 6). See also Sec. 5.A, where passivity is shown. The entire control system described in Sec. 5.4 is given

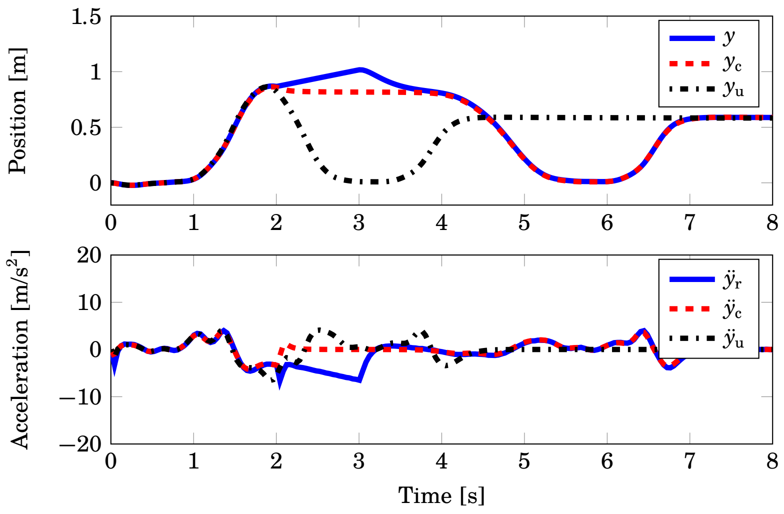


Figure 5.6 Similar to Fig. 5.5, except that y was moved away from the intended path between 2 s and 3 s. Again, the behavior was satisfactory both regarding position and acceleration.

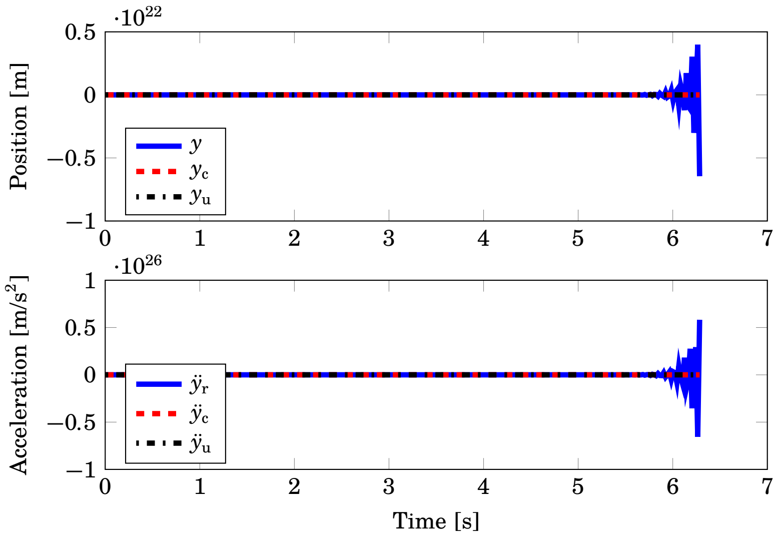


Figure 5.7 Using the control system in [Ijspeert et al., 2013], subject to the simulated noise and time delay, resulted in unstable behavior.

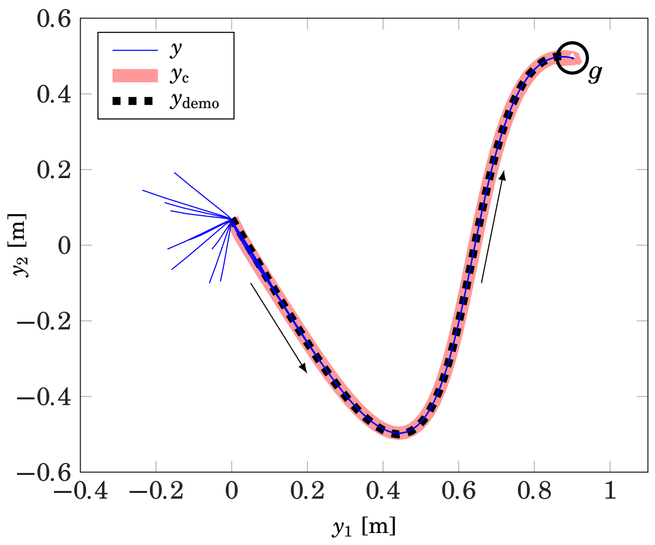


Figure 5.8 Positions of simulated trajectories. The movement direction is indicated by the arrows. For an uncluttered view, only 10 of the 100 simulated trajectories are displayed. For each simulation, y started to the left in the plot, with some distance to y_c. Then, y moved toward y_c. Subsequently, both y and y_c converged to the goal position g, which is marked with a circle at the upper right of the plot. Since y_c was initialized to the same point for each simulation, it always followed the same path. Further, it followed the demonstrated path closely, which is expected given a sufficient number of basis functions ($N_b = 50$ were used).

by

$$\ddot{y}_r = k_p(y_c - y) + k_v(\dot{y}_c - \dot{y}) + \ddot{y}_c \tag{5.12a}$$
$$\dot{e} = \alpha_e(y - y_c - e) \tag{5.12b}$$
$$\tau_a = \tau(1 + k_c e^T e) \tag{5.12c}$$
$$\tau_a \dot{x} = -\alpha_x x \tag{5.12d}$$
$$\tau_a \dot{y}_c = z \tag{5.12e}$$
$$\tau_a \dot{z} = \alpha(\beta(g - y_c) - z) + f(x) \tag{5.12f}$$

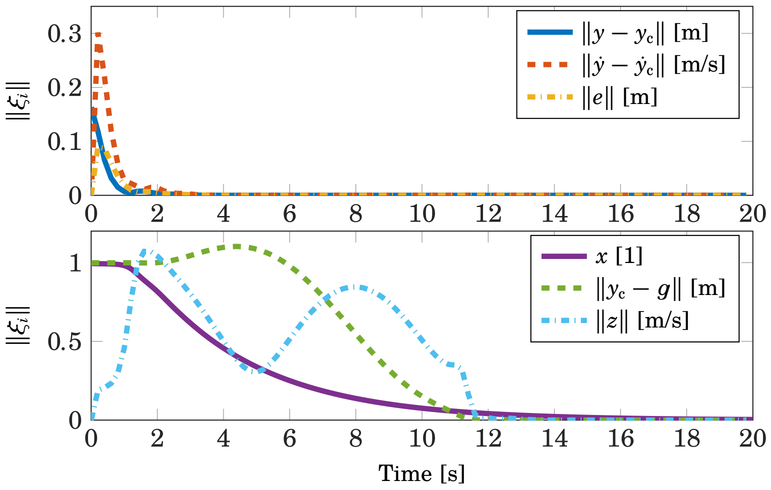


Figure 5.9 The magnitudes of DMP states ξ_i plotted over time, for one of the simulations displayed in Fig. 5.8. The notation $\|\cdot\|$ represents the 2-norm, and the unit symbol [1] indicates dimensionless quantity. It can be seen that y converged to y_c, and that e converged to 0. As will be shown in Sec. 5.6, this is predicted by Theorem 5.1. Further, each state converged to 0, which is predicted by Theorem 5.2.

We introduce the state vector ξ as

$$\xi = \begin{pmatrix} y - y_c \\ \dot{y} - \dot{y}_c \\ e \\ x \\ y_c - g \\ z \end{pmatrix} \tag{5.13}$$

and write the system on state-space form,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y - y_c \\ \dot{y} - \dot{y}_c \\ e \\ x \\ y_c - g \\ z \end{pmatrix} = \begin{pmatrix} \dot{y} - \dot{y}_c \\ -k_p(y - y_c) - k_v(\dot{y} - \dot{y}_c) \\ \alpha_e(y - y_c - e) \\ -\dfrac{\alpha_x}{\tau_a}x \\ \dfrac{1}{\tau_a}z \\ \dfrac{\alpha}{\tau_a}(\beta(g - y_c) - z) + \dfrac{1}{\tau_a}f(x) \end{pmatrix} \tag{5.14}$$

The states $y - y_\mathrm{c}$, $\dot{y} - \dot{y}_\mathrm{c}$, and e have the same dimension, which we denote by n. Let I_n be the identity matrix, and 0_n the zero matrix, each of size n. The upper part of (5.14) is linear and can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ e \end{pmatrix} = \begin{pmatrix} 0_n & I_n & 0_n \\ -k_\mathrm{p} I_n & -k_\mathrm{v} I_n & 0_n \\ \alpha_e I_n & 0_n & -\alpha_e I_n \end{pmatrix} \begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ e \end{pmatrix} \tag{5.15}$$

Denote by $\bar{\xi}$ and $\bar{A}$ the state vector and the system matrix in (5.15), respectively.

Theorem 5.1
The dynamical system defined by (5.15) for DMP operation has $\bar{\xi} = 0$ as a globally exponentially stable equilibrium. □

Proof. For $\bar{\xi} = 0$ we have $\mathrm{d}\bar{\xi}/\mathrm{d}t = 0$ in (5.15), so $\bar{\xi} = 0$ is an equilibrium. The system is globally asymptotically stable if the real part of each eigenvalue of $\bar{A}$ is strictly negative [Glad and Ljung, 2000]. With $k_\mathrm{p} = k_\mathrm{v}^2/4$ (see Sec. 5.4) the eigenvalues are given by

$$\lambda_{1,\dots,2n} = -\frac{k_\mathrm{v}}{2} \tag{5.16a}$$

$$\lambda_{2n+1,\dots,3n} = -\alpha_e \tag{5.16b}$$

These are strictly negative, since $k_\mathrm{v}, \alpha_e > 0$. The linear system is hence globally asymptotically stable. For linear systems, asymptotic stability is equivalent with exponential stability [Slotine and Li, 1991; Rugh, 1996]. The system is therefore globally exponentially stable. □

Next, we will show that the system given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ e \\ x \end{pmatrix} = \begin{pmatrix} \dot{y} - \dot{y}_\mathrm{c} \\ -k_\mathrm{p}(y - y_\mathrm{c}) - k_\mathrm{v}(\dot{y} - \dot{y}_\mathrm{c}) \\ \alpha_e(y - y_\mathrm{c} - e) \\ -\dfrac{\alpha_x}{\tau_\mathrm{a}} x \end{pmatrix} \tag{5.17}$$

is contracting, of which a definition can be found in [Lohmiller and Slotine, 1998]. We note that this is a hierarchical system, consisting of (5.15), which does not depend on x, and of $\dot{x} = -\alpha_x x/\tau_\mathrm{a}$. To show contraction, we will use the following proposition.

Proposition 5.1
If $\dot{x}_1 = g_1(x_1)$ is contracting, and $\dot{x}_2 = g_2(x_1, x_2)$ is contracting for each fixed x_1, then the hierarchy

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} g_1(x_1) \\ g_2(x_1, x_2) \end{pmatrix} \tag{5.18}$$

is contracting. □

This follows directly from Proposition 2 in [Wensing and Slotine, 2017], by applying it to autonomous systems.

Proposition 5.2
The system given by (5.17) is contracting. □

Proof. We know from Theorem 5.1 that (5.15) is globally exponentially stable. It is therefore contracting [Lohmiller and Slotine, 1998]. For the fixed point $\bar{\xi} = 0$, we have $\dot{x} = -\alpha_x x/\tau$, which is contracting since $-\alpha_x/\tau < 0$. It now follows from Proposition 5.1 that the hierarchical combination (5.17) is contracting. □

We now address the stability of the entire control system in (5.14) as follows.

Theorem 5.2
The system given by (5.14) for DMP operation has $\xi = 0$ as a globally exponentially stable equilibrium point. □

Proof. Since x is a factor of $f(x)$ and the remaining part of $f(x)$ is bounded, $f(0) = 0$, see Chapter 3 and [Ijspeert et al., 2013]. It can therefore be seen that $\xi = 0$ yields $\mathrm{d}\xi/\mathrm{d}t = 0$, so $\xi = 0$ is an equilibrium point. It remains to show the stability. We know from Proposition 5.2 that (5.17) is contracting, and it can be seen that it has the origin as a fixed point. Consider now the remaining part of (5.14), *i.e.*,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y_\mathrm{c} - g \\ z \end{pmatrix} = \begin{pmatrix} \dfrac{1}{\tau_\mathrm{a}} z \\ \dfrac{\alpha}{\tau_\mathrm{a}}(\beta(g - y_\mathrm{c}) - z) + \dfrac{1}{\tau_\mathrm{a}} f(x) \end{pmatrix} \tag{5.19}$$

For the fixed point of (5.17) we have $\tau_\mathrm{a} = \tau$ and $f(x) = f(0) = 0$, and the system (5.19) then simplifies to

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y_\mathrm{c} - g \\ z \end{pmatrix} = \begin{pmatrix} 0_n & \dfrac{1}{\tau} I_n \\ -\dfrac{\alpha\beta}{\tau} I_n & -\dfrac{\alpha}{\tau} I_n \end{pmatrix} \begin{pmatrix} y_\mathrm{c} - g \\ z \end{pmatrix} \tag{5.20}$$

This system is linear, and with $\beta = \alpha/4$ (see Chapter 3) the eigenvalues of the system matrix are given by

$$\mu_{1,\dots,2n} = -\frac{\alpha}{2\tau} \tag{5.21}$$

Since the eigenvalues are strictly negative, the system (5.20) is contracting. Further, we note that (5.14) is hierarchical from (5.17) to (5.19). Therefore, it follows from Proposition 5.1 that (5.14) is contracting. This is true for the whole state space. Hence, all solutions of (5.14) converge exponentially to the same trajectory. Since one solution is given by $\xi = 0$, they must all converge exponentially to this equilibrium point. Thus, (5.14) is globally exponentially stable. □

Since $y = g$ for $\xi = 0$, Theorem 5.2 implies that the system (5.14) converges exponentially to a state where the goal configuration is reached by the actual robot position.

5.7 Experiments

The algorithm in Sec. 5.4 was implemented on the ABB YuMi prototype robot, see Chapter 2. Similar to the simulations, the control system ran at 250 Hz, and the delay between robot and controller was 3 sample periods, corresponding to 12 ms. The computations took place in joint space, and the robot's forward kinematics were used for visualization in Cartesian space in some of the figures presented.

Six different setups were used to evaluate the control algorithm. For each setup, 50 trials were made, yielding a total of 300 trials. Prior to each trial, a temporally coupled DMP had been determined from demonstration by means of lead-through programming [Stolt et al., 2015a]. In each trial, the temporally coupled DMP was executed while data were logged and saved. Perturbations were introduced by physical contact with a human. A wrist-mounted ATI Mini40 force/torque sensor was used to measure the contact force, and a corresponding acceleration was added to $\ddot{y}_{\mathrm{r}}$ as a load disturbance. The purpose of Setups 5.1 and 5.2 was to demonstrate the proposed functionality in general and examine the reference acceleration, whereas the main purpose of Setups 5.3 to 5.6 was to verify the exponential stability shown in Sec. 5.6. The six setups were as follows.

Setup 5.1 This setup is visualized in Fig. 5.10. The task of the robot was to place a stop button in its corresponding case. The assembly parts are shown in Fig. 5.11. The human introduced two perturbations during the DMP execution. The first perturbation was formed by moving the end-effector away from its path, and then releasing it. The second perturbation consisted of a longer, unstructured, movement later along the trajectory.

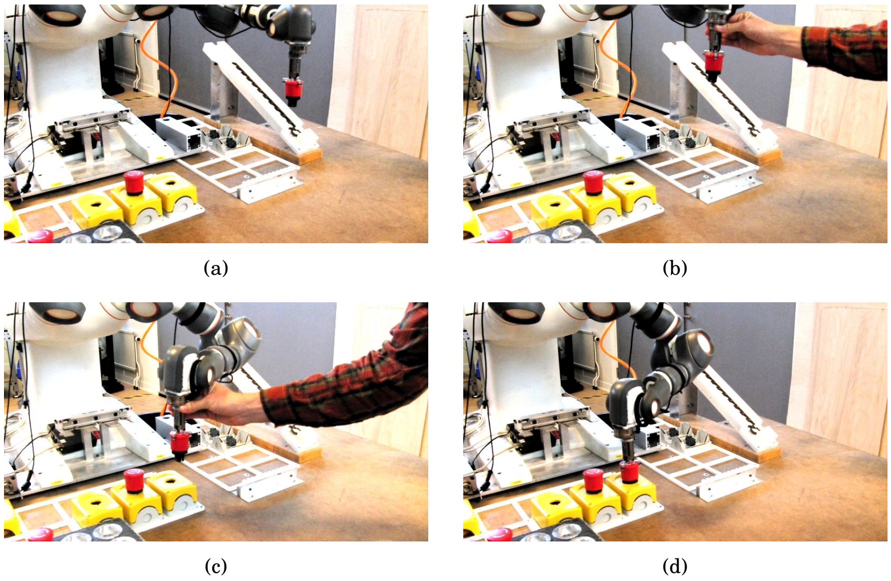


Figure 5.10 Setup 5.1. In (a), the robot started to execute a DMP for placing the stop button in the rightmost yellow case. A human perturbed the motion twice. The first perturbation (b) was formed by moving the end-effector away from its path, and then releasing it. The second perturbation (c) lasted for a longer time, and consisted of unstructured movement. The robot recovered from both perturbations, and managed to place the stop button in the case (d). Data from one trial are shown in Fig. 5.15.

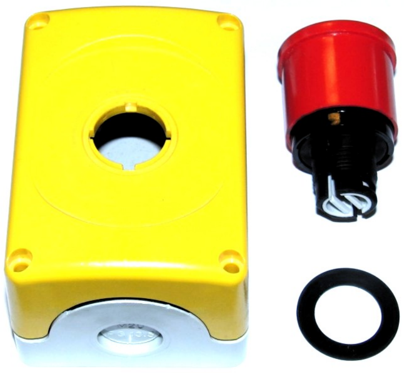


Figure 5.11 Yellow case (left), stop button (upper right) and gasket (lower right) used in Setups 5.1, 5.2 and 5.6.

Setup 5.2 This setup is visualized in Fig. 5.12. A human co-worker realized that the stop buttons in the current batch were missing rubber gaskets, and acted to modify the robot trajectory, allowing for the co-worker to attach the gasket on the stop button manually. During execution of the DMP, the end-effector was stopped and lifted to a comfortable height by the co-worker. Thereafter, the gasket was attached, and finally the end-effector was released. For the sake of completeness, the modified trajectory was used to form yet another DMP, which allowed for the co-worker to attach the gaskets without perturbing the trajectory of the robot, for the remaining buttons in the batch. To verify this functionality, one such modified DMP was executed at the end of each trial.

Setup 5.3 The robot moved one of its arms without manipulating any objects. The arm was subjected to two perturbations.

Setup 5.4 The robot moved both of its arms simultaneously, again without any manipulation. Two perturbations were introduced: first one on the left arm, and subsequently one on the right arm.

Setup 5.5 This setup is visualized in Fig. 5.13. The robot used both arms to pick up a ball. One perturbation was introduced on the left arm.

Setup 5.6 This setup is visualized in Fig. 5.14. The objective of the robot was to mate a stop button, grasped with the right arm, with its corresponding case, grasped with the left arm. The left arm was subjected to one perturbation.

Videos

A video of Setups 5.1 and 5.2 is available in [Karlsson, 2016]. Further, a video of Setups 5.3 to 5.6 is available in [Karlsson, 2017a]. In Part I of the latter video, the experimental setups are shown, except that the temporally coupled DMPs were executed without any perturbations present. This is to visualize what should be achieved by the robot, in each setup. In Part II, DMPs without temporal coupling were used, and the robot was subject to perturbations. The purpose of this part is to demonstrate that the robot risks to fail in such a scenario, and hence motivate why temporal coupling is necessary. This risk of failure was also indicated in [Ijspeert et al., 2013], using one-dimensional simulation examples. Part III of the video shows the experiments, where temporally coupled DMPs were executed in the presence of perturbations.

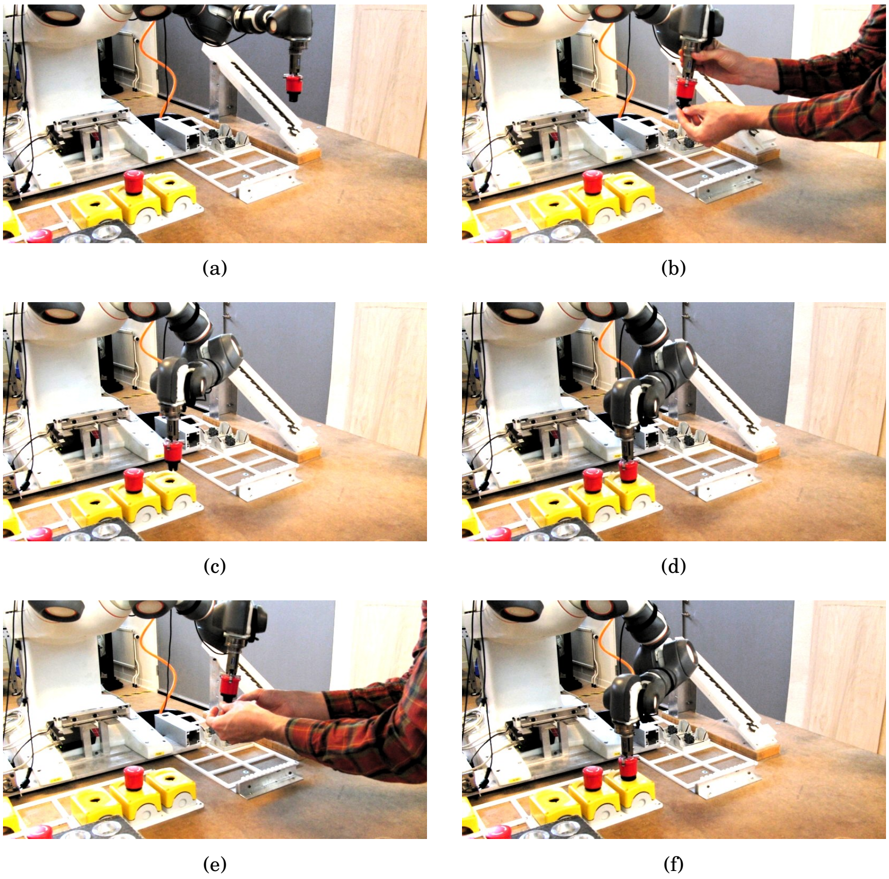


Figure 5.12 Setup 5.2. The robot started its motion toward the rightmost yellow case in (a). The end-effector was stopped and lifted, and the gasket was mounted in (b). The robot was then released, and continued its motion to the case, (c) and (d). The actual trajectory was saved and used to form a modified DMP, and the robot was reset to a configuration similar to that in (a). When executing the modified DMP, the human co-worker could attach the gasket without perturbing the motion of the robot (e). The robot finished the modified DMP in (f). Data from one trial are shown in Fig. 5.16.

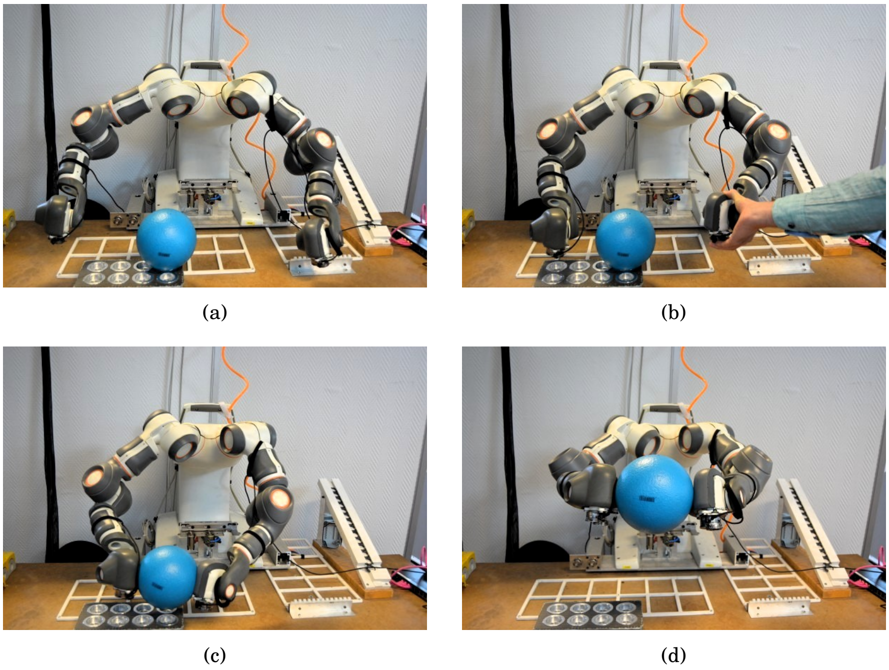


Figure 5.13 Setup 5.5. The robot task was to pick up the blue ball using both arms. The movement started in (a), and a human perturbed the movement in (b). Thereafter, the robot was released and recovered from the perturbation. The arms reached the ball simultaneously in (c). The goal state was then reached in (d). Data from one trial are shown in Fig. 5.19.

5.8 Results

In this section, data from one trial of each of the setups in Sec. 5.7 are displayed. For each setup, all trials gave qualitatively similar results. In each of the 300 trials, the perturbations were first recovered from and thereafter the goal state was reached.

Data from a trial of Setup 5.1 are displayed in Fig. 5.15. As intended, the two disturbances were successfully recovered from. The reference acceleration was of realizable magnitude.

Data from a trial of Setup 5.2 are displayed in Fig. 5.16. First, the perturbation was successfully recovered from as intended. The reference acceleration was of realizable magnitude. When the modified DMP was executed, it behaved like a smooth version of the perturbed original trajectory.

Data from Setups 5.3 to 5.6 are shown in Figs. 5.17 to 5.20. The figures show the magnitudes of the states in (5.14) represented in joint space, plotted over time. The perturbations are clearly visible in the data. It can be

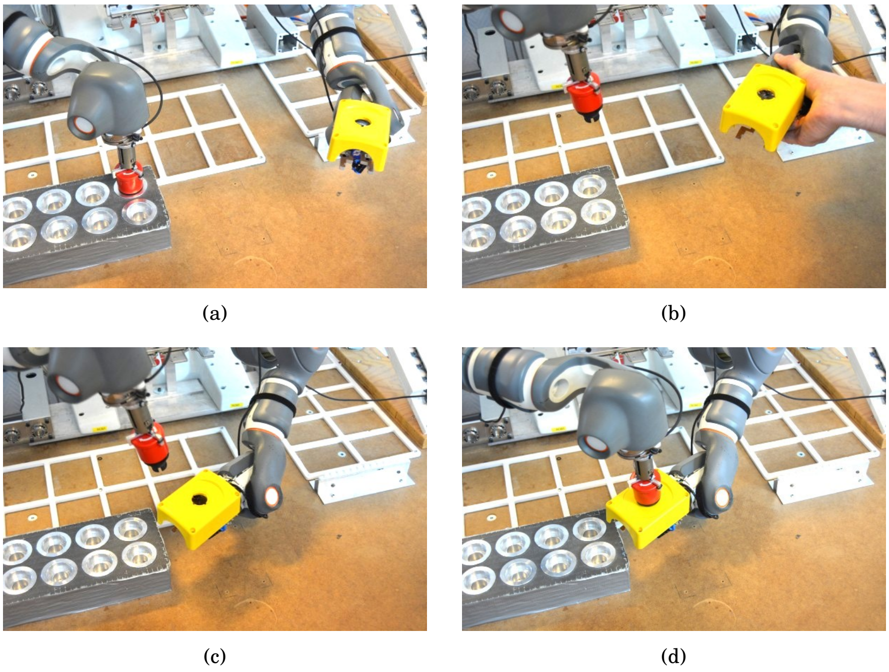


Figure 5.14 Setup 5.6. The task was to insert the red stop button into the corresponding hole in the yellow case. The initial configuration is shown in (a). In (b), a human perturbed the movement, and in (c) the robot had recovered from the perturbation. The goal configuration was reached in (d). Data from one trial are shown in Fig. 5.20.

seen that y converged to y_c, and that e converged to 0, after each perturbation. This is expected from Theorem 5.1. The data also show that each state converged to 0, as predicted by Theorem 5.2.

5.9 Discussion

Compared to the previous research in [Ijspeert et al., 2013], the method in this chapter contains the following extensions. Feedforward control was added to the PD controller, thus forming a two-degree-of-freedom controller. Further, the PD controller gains were reduced to moderate magnitudes. The expression for τ_a was also modified, to include the original time constant τ as a factor. These changes resulted in the following benefits, compared to the previous method. The feedforward part allowed for the controller to act also for insignificant position and velocity error, thus improving the trajectory tracking. Because of this, the large controller gains used in previous

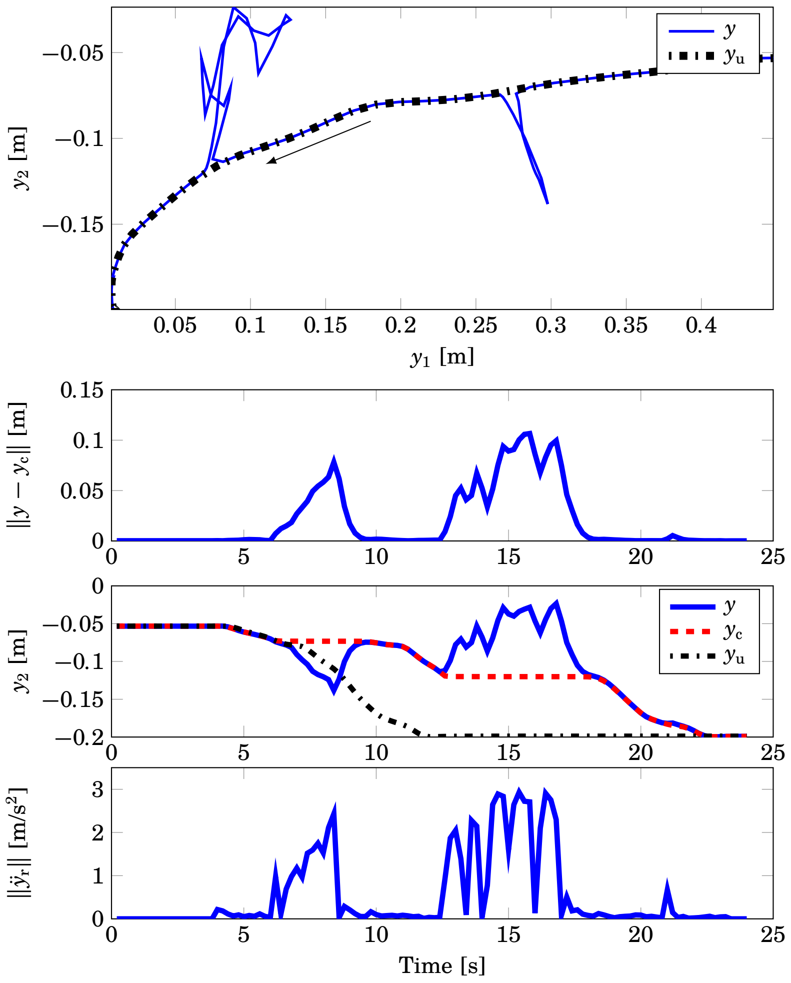


Figure 5.15 Experimental data from a trial of Setup 5.1. The first (from above) plot shows two position coordinates, y_1 and y_2, of the end-effector in Cartesian space. The arrow indicates the direction of the movement, which started in the upper right and finished in the lower left of the plot. The two perturbations are clearly visible. The second plot shows the distance between y and y_c over time. In the third plot, it can be seen that the evolution of y_c slowed down during each perturbation. Subsequently, y recovered, and when it was close to y_c, the movement continued as a delayed version of y_u. The reference acceleration was of realizable magnitude, as shown in the fourth plot.

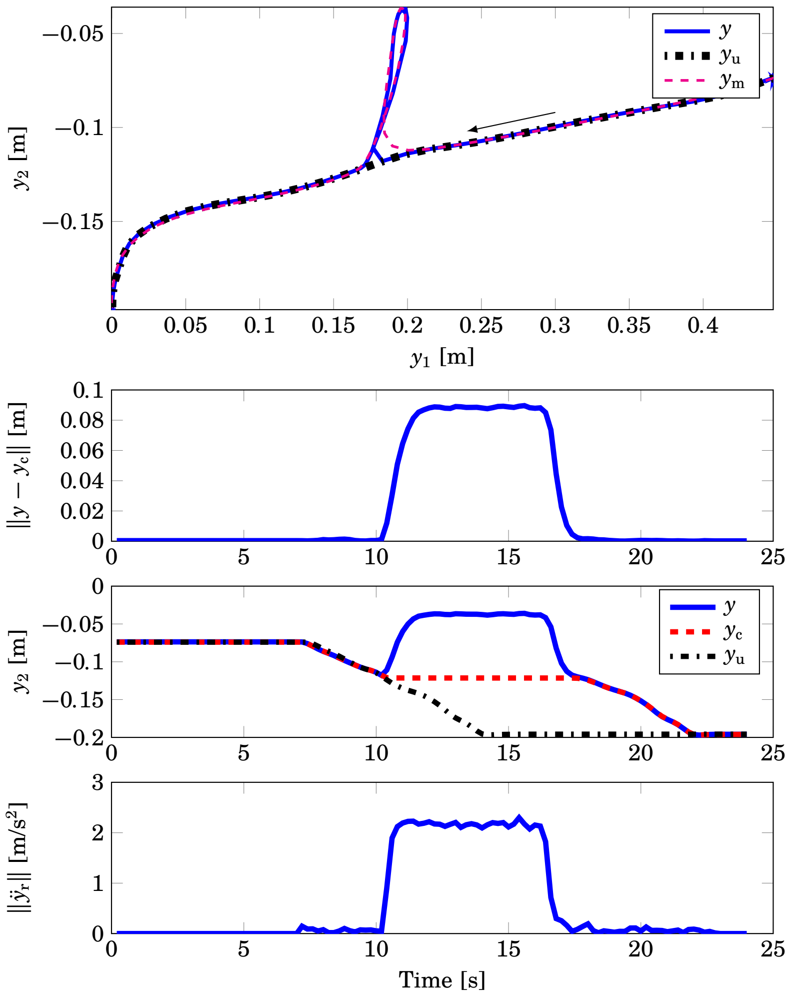


Figure 5.16 Experimental data from a trial of Setup 5.2. The organization of this figure is similar to that of Fig. 5.15. The perturbation for stopping and lifting the end-effector took place from time 10 s to 17.5 s, and is clearly visible in each plot. This perturbation was recovered from as intended, and the reference acceleration was of realizable magnitude. The uppermost plot also displays the measured trajectory obtained by executing the modified DMP, denoted y_m. It behaved like a smooth version of the perturbed original trajectory y.

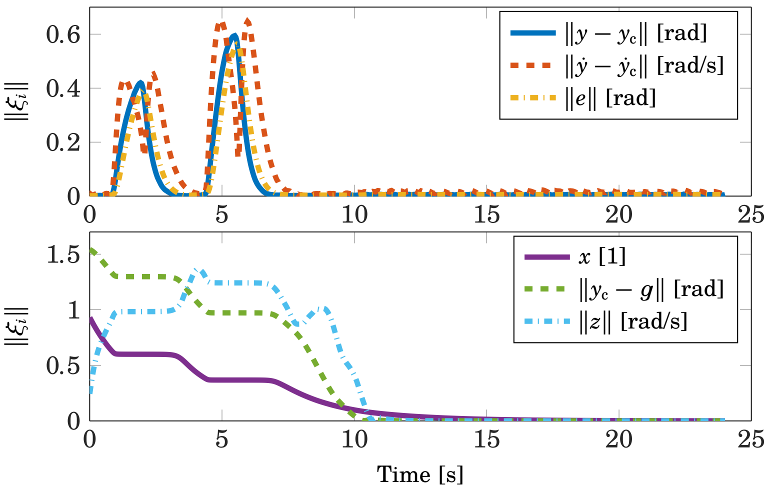


Figure 5.17 Data from a trial of Setup 5.3, where the left arm was perturbed twice during its movement.

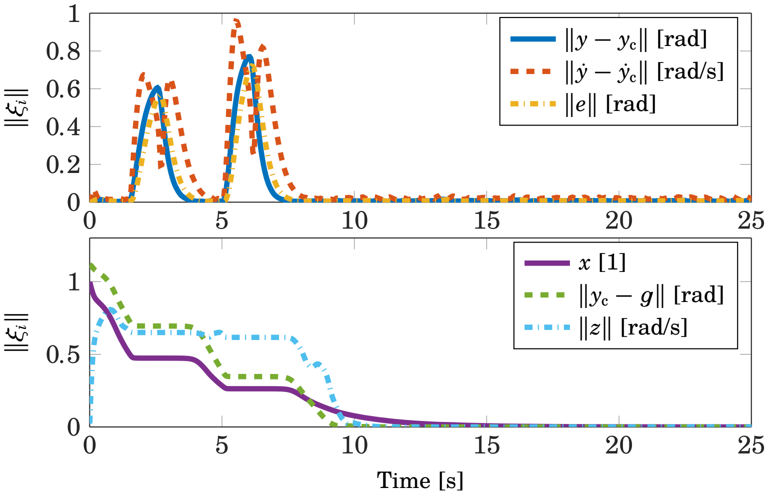


Figure 5.18 Data from a trial of Setup 5.4, where both arms moved simultaneously and were perturbed once each.

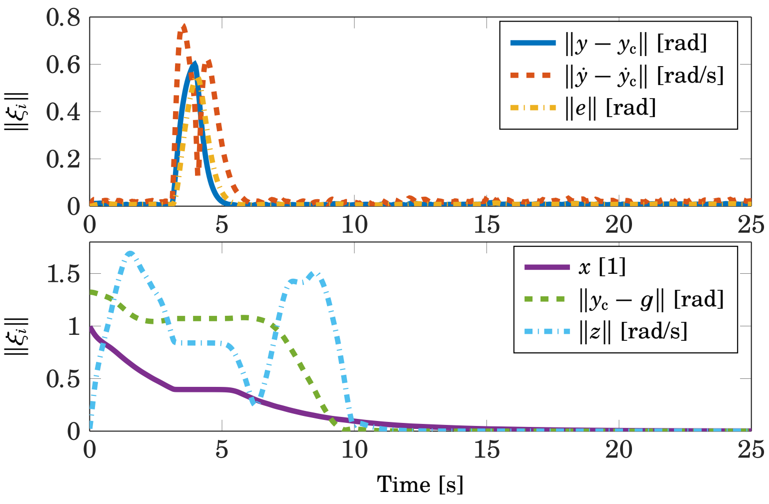


Figure 5.19 Data from a trial of Setup 5.5. The setup is shown in Fig. 5.13.

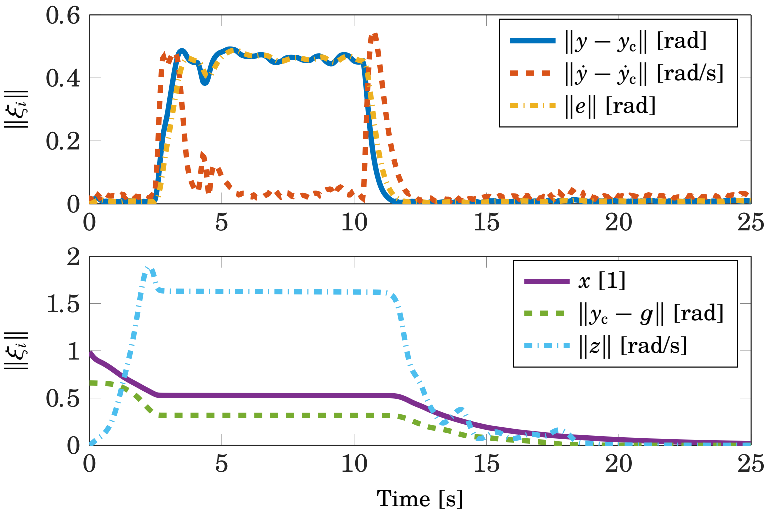


Figure 5.20 Data from a trial of Setup 5.6. The setup is shown in Fig. 5.14. In this particular trial, the robot was grasped by the human for a significantly longer time than in the trials shown in Figs. 5.17 to 5.19, though this was not the case for all trials of Setup 5.6.

research, that were used to mitigate significant tracking errors, could be reduced to moderate magnitudes. In turn, using moderate gains instead of very large ones, resulted in practically realizable control signals, instead of prohibitively large ones. It also improved the delay margin significantly. The modification of the expression for τ_a was not the main focus of this research, but it was necessary since it allowed for the actual trajectory to evolve according to the DMP, with time constant τ. Without this modification, the time parameter τ would not have affected the trajectory generated by the DMP. Instead, τ_a would have converged to 1, regardless of τ, which would not have been desirable.

The work presented here focused on the control structure for trajectory tracking and perturbation recovery, rather than on the perturbations themselves. Even though the perturbations in the experiments considered here emerged from physical contact with a human, the control structure would work similarly for any type of perturbation. There are many other possible perturbations, *e.g.*, a pause of the movement until a certain condition is fulfilled, superpositioned motion-control signals to explore the surroundings with a force/torque sensor, a detour to enable line of sight between a camera and a part of the work space, exploration in reinforcement learning, or any other unforeseen deviation from the reference trajectory defined by a DMP.

Already in the original DMP versions, without temporal coupling, a phase variable x is used instead of time [Ijspeert et al., 2013; Pastor et al., 2013]. This has been motivated by easier resetting, and by results that suggest that some biological systems do not have access to time; see, *e.g.*, [Keating and Thach, 1997]. However, without temporal coupling, x evolves deterministically according to (5.2). Solving the differential equation yields

$$x = x_0 \exp\Big(-\frac{\alpha_x}{\tau} t\Big) \tag{5.22}$$

where t represents time and the initial value x_0 could be set to 1; see Chapter 2. Hence, an expression with explicit time could be used instead of x; see also [Chen et al., 2016] where explicit time was used to determine the virtual forcing term. However, this is not as straightforward when temporal coupling is included. The evolution rate of real time can not be controlled, and some virtual time with variable time derivative would have to be used. The phase variable x takes the role of such virtual time. Hence, one could argue that the phase variable is truly justified only for temporally coupled DMPs.

The coupling term C_t has been introduced in previous research to drive y_c toward y when these were different, see Sec. 5.2. However, whether this effect is desired, and to what extent, is context dependent. Further, the effect would be mitigated by the temporal coupling, which would slow down the evolution of y_c in (5.6). Which of these effects that would be dominant in different cases would be difficult to predict intuitively. For these two reasons, the coupling term was not included in the method proposed here, though it would be straightforward to implement. It was, however, included in the simulations where the previous method, described in Sec. 5.2, was evaluated. During the perturbations in Figs. 5.1 and 5.2, the effect of the temporal coupling was dominant, as y_c did not approach y significantly.

It is important to note that the 'Robot' block in Fig. 5.3 includes both the robot itself and the internal robot controller. It is possible for the internal control loop to achieve a reference acceleration, $\ddot{y}_r$, with small error, as long as $\ddot{y}_r$ is reasonably smooth and limited in magnitude. Therefore, modeling the robot as a double integrator can be justified by feedback linearization [Slotine and Li, 1991] even though the dynamics of the robot itself (*i.e.*, excluding internal controller, *e.g.*, from joint torques to joint angles) are more complicated than a double integrator.

A common approach to prove stability of a nonlinear system is to find a Lyapunov function of the state vector. In Sec. 5.A, it is shown that this is not straightforward for the control system considered here. Therefore, instead the contraction theory in [Lohmiller and Slotine, 1998] was used in this chapter. Some care must be taken when analyzing stability based on contraction theory, because contraction alone does not imply stability. For instance, the following system has been given in [Lohmiller and Slotine, 1998] as an example of a contraction.

$$\dot{x}_1 = -x_1 + \exp(t) \tag{5.23}$$

Here, x_1 represents a scalar state and t is time. The system is indeed a contraction because the Jacobian is -1 and hence uniformly negative definite [Lohmiller and Slotine, 1998]. However, one solution is given by

$$x_1 = \frac{\exp(t)}{2} \tag{5.24}$$

which is not stable. It can be seen that $x_1 \to \infty$ as $t \to \infty$. The contraction guarantees exponential convergence to a single trajectory [Lohmiller and Slotine, 1998], in this case to (5.24), but it does not guarantee exponential convergence to a constant. Nevertheless, for autonomous systems such as DMPs, global contraction is equivalent with global exponential stability [Lohmiller and Slotine, 1998].

Even though Theorem 5.2 states that the state vector ξ converges to the origin, in practice noise and model error prevent the state vector from staying arbitrarily close to the origin. In the experiments there was an unmodeled time delay of 12 ms between the controller in (5.8) and the internal robot controller, which is an example of a model error. Further, the robot end-effectors reached their targets with an accuracy of ± 1 mm.

Because the origin could not be reached by ξ with exactly zero error, the movement could for instance be considered finalized once $\|\xi\| < \rho$ for some small constant ρ, or upon force-based detection of task completion as in Chapter 7. A related discussion about convergence toward single-point goals can be found in [LaValle, 2006]. There, the concept of so-called asymptotic solution plans is also described. Since exponential stability is a stronger requirement than asymptotic stability [Slotine and Li, 1991], Theorem 5.2 implies that temporally coupled DMPs are asymptotic solution plans.

Stability for the original DMP framework was shown in [Perk and Slotine, 2006; Wensing and Slotine, 2017] by utilizing that $f(x)$ converged to 0, which followed from the fact that x decayed exponentially, regardless of any deviation from the intended movement. However, this is true only if temporal coupling is not used, and the convergence of $f(x)$ is less obvious for the DMP version studied in this chapter. Due to the adaptive time parameter τ_a in (5.4), it can not be assumed that x decays exponentially for temporally coupled DMPs. Instead, the stability of temporally coupled DMPs has now been established in the proof of Theorem 5.2.

In the experiments, a force sensor was mounted on each wrist of the robot, and the measured forces were scaled and added to the reference accelerations as disturbances, which allowed for the human to introduce perturbations in an intuitive way. This arrangement worked well for the sake of evaluating the stability properties of temporally coupled DMPs, and it was straightforward to implement. However, for the purpose of interacting physically with the robot, one could also consider passive force control. Such an approach was presented in [Denisa et al., 2016], though temporal coupling was not included. As suggested in [Denisa et al., 2016], it would therefore be interesting to combine temporal coupling with passive force control in the DMP framework in future research.

The effect of temporal coupling can be seen both in the simulated and the experimental data. For example, consider Fig. 5.20 where the robot was grasped by the human at $t \approx 2.5$ s. During the perturbation, the evolution of x, y_c, and z was slowed down. Such behavior is desired in general, since it helps to retain the coordination between the degrees of freedom of the robot. It resulted from the fact that e had a significant magnitude, which in turn caused τ_a to be significantly larger than τ, see (5.9). The robot was released at $t \approx 10$ s, and the controller described by (5.8) started to drive y to y_c, whereby e was driven to 0. After e had converged to insignificant magnitude, at approximately 12 s, it can be seen that x, y_c, and z evolved considerably faster. The same phenomenon could be seen in each experiment, as well as in the simulations.

It is necessary to implement saturation on the control signals, to prevent too large acceleration and velocity for large perturbations. Such boundaries were implemented, but never reached in the experiments in this work.

In the current implementation, the actual trajectory returned to the reference trajectory, approximately where it started to deviate. This might not always be desired. For instance, it might sometimes be more practical to connect further along the reference trajectory, *e.g.*, after avoiding an obstacle. A lower value of k_c would result in such behavior. However, it must then be known what value of k_c that should be used. Further, one could think of scenarios where it would not be desirable to connect to the reference trajectory, *e.g.*, if a human would modify the last part of the trajectory to a new end point. Hence, future work includes development of a method to determine the desired behavior after a perturbation. The method presented in this chapter would be useful for executing the desired behavior, once it could be determined. Nevertheless, one can think of various scenarios where the recovery presented here would be desirable, such as those in Sec. 5.7.

The proposed control algorithm in Sec. 5.4 includes some parameters that should be chosen, but these have intuitive meanings. To avoid overshoot, the control gains in (5.8) should be constrained as described in Sec. 5.4. This yields a negative real double pole for the system (5.8), located in $-k_v/2$, and a larger magnitude of the double pole corresponds to a faster control system. Similarly, α and β should be constrained as described in Chapter 3 and [Ijspeert et al., 2013]. The control system is intentionally compliant to unforeseen disturbances, such as contact with a human, and therefore no integral parts are used in the control loops that operate outside the internal robot controller. Introducing integral parts could also yield overshoot in the position domain, which in the context of robotics might result in collision. The constant α_e determines the extent of low-pass filtering from position error to e, and k_c determines how much the DMP evolution should slow down in presence of position deviations.

Part II of the video in [Karlsson, 2017a] shows Setups 5.3 to 5.6, but the original DMP formulation, without temporal coupling, was used instead. Setups 5.5 and 5.6 were deliberately designed to require coordination between both arms, in order to highlight the necessity of temporal coupling for DMPs. Without temporal coupling, the coordination was lost in presence of any significant perturbation, and hence these tasks failed in Part II. In fact, the coordination between all degrees of freedom of the robot, and not only between the two arms, might be lost. For instance, only the left arm moved in Setup 5.1, and yet the behavior was not satisfactory in Part II of the video. It can also be noted that the coordination was retained in Part III, despite perturbations, thanks to the temporal coupling.

In the experiments, the DMPs were defined in joint space. The implementation in joint space allowed for the operator to specify the entire configuration on each seven-degree-of-freedom robot arm, rather than just the tool pose. The DMPs could also have been defined in Cartesian space, see for instance [Ijspeert et al., 2013; Ude et al., 2014]. However, whereas the assumption of a contractible state space in Sec. 5.6 is correct for the robot joint space and the position in Cartesian space, some care must be taken regarding the orientation in Cartesian space, as discussed in, *e.g.*, [Mayhew et al., 2011]. Restrictions on the orientation must be invoked to guarantee that it is contained in a contractible space. Finding such restrictions, that are not unnecessarily conservative, is the subject of Chapter 6.

5.10 Conclusion

In this research, it was shown how perturbations of DMPs could be recovered from, while preserving the characteristics of the original DMP framework in the absence of significant perturbations. Feedforward control was used to track the reference trajectory generated by a DMP. Feedback control with moderate gains was used to suppress deviations. This design is the first, to the best of the author's knowledge, that takes the following aspects into account. In the absence of significant disturbances, the position error must be small enough, so that the dynamical system would not slow down unnecessarily due to the temporal coupling. Very large controller gains would result in small errors under ideal conditions, but in practice they yield control signals that are not realizable. On the other hand, if the gains are moderate and only feedback control is used, too large errors occur.

Feedforward enabled the controller to act even without significant error, which in turn allowed for moderate controller gains. The proposed control system was verified in simulations and experimentally. Videos of the experiments are available in [Karlsson, 2016; Karlsson, 2017a]. Further, it was shown mathematically that the proposed control system is globally exponentially stable, which implies exponential convergence to a steady state in which the goal position is reached by the actual robot position.

Appendix A. Passivity of Temporally Coupled Dynamical Movement Primitives

We here consider a passivity property of temporally coupled DMPs. The passivity formalism is described in, *e.g.*, [Slotine and Li, 1991; Khalil, 2002]. The entire control system for temporally coupled DMPs in Sec. 5.6 is given below for convenience.

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ e \\ x \\ y_\mathrm{c} - g \\ z \end{pmatrix} = \begin{pmatrix} \dot{y} - \dot{y}_\mathrm{c} \\ -k_\mathrm{p}(y - y_\mathrm{c}) - k_\mathrm{v}(\dot{y} - \dot{y}_\mathrm{c}) \\ \alpha_e(y - y_\mathrm{c} - e) \\ -\frac{\alpha_x}{\tau_\mathrm{a}} x \\ \frac{1}{\tau_\mathrm{a}} z \\ \frac{\alpha}{\tau_\mathrm{a}}(\beta(g - y_\mathrm{c}) - z) + \frac{1}{\tau_\mathrm{a}} f(x) \end{pmatrix} \tag{5.25}$$

THEOREM 5.3
Consider the system defined by (5.25) for DMP operation. The virtual force $f(x)$ can be seen as an external input to the system. The mapping from $f(x)$ to z/τ_a is passive. □

Proof. The system is passive if $f(0) = 0$, $\dot{\xi} = 0$ for $\xi = f(x) = 0$, and there exists a C^1 function $V : \mathbb{R}^{5n+1} \to \mathbb{R}$, called a storage function, for which the following criteria hold [Khalil, 2002].

$$V(0) = 0 \tag{5.26a}$$
$$V(\xi) \geq 0, \quad \forall \xi \neq 0 \tag{5.26b}$$
$$\dot{V}(\xi) \leq f(x)^\mathrm{T} z/\tau_\mathrm{a}, \quad \forall \xi, f(x) \tag{5.26c}$$

Since x is a factor of $f(x)$ and the remaining part of $f(x)$ is bounded, $f(0) = 0$, see Chapter 3 and [Ijspeert et al., 2013]. It can further be seen that $\dot{\xi} = 0$ for $\xi = f(x) = 0$.

The states $y - y_\mathrm{c}$, $\dot{y} - \dot{y}_\mathrm{c}$, and e have the same dimension, which we denote by n. Let I_n be the identity matrix, and 0_n the zero matrix, each of size n. The upper part of (5.25) is given by the following linear system.

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ e \end{pmatrix} = \begin{pmatrix} 0_n & I_n & 0_n \\ -k_\mathrm{p} I_n & -k_\mathrm{v} I_n & 0_n \\ \alpha_e I_n & 0_n & -\alpha_e I_n \end{pmatrix} \begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ e \end{pmatrix} \tag{5.27}$$

Denote by $\bar{\xi}$ and $\bar{A}$ the state vector and the system matrix in (5.27), respectively. Because $k_\mathrm{p} = k_\mathrm{v}^2/4$ (see Sec. 5.4) the eigenvalues are given by

$$\lambda_{1,\dots,2n} = -\frac{k_\mathrm{v}}{2} \tag{5.28a}$$
$$\lambda_{2n+1,\dots,3n} = -\alpha_e \tag{5.28b}$$

Since k_v, $\alpha_e > 0$, the eigenvalues are strictly negative. Let Q be a symmetric, positive definite matrix of size $3n$. Since $\bar{A}$ has all eigenvalues strictly in the left half-plane, it follows [Glad and Ljung, 2000] that there exists a symmetric, positive definite matrix P solving the Lyapunov equation

$$P\bar{A} + \bar{A}^{\mathrm{T}}P = -Q \tag{5.29}$$

Now, assume the following storage function V.

$$V(\xi) = \bar{\xi}^{\mathrm{T}}P\bar{\xi} + \frac{1}{2}x^2 + \frac{1}{2}\alpha\beta(y_{\mathrm{c}} - g)^{\mathrm{T}}(y_{\mathrm{c}} - g) + \frac{1}{2}z^{\mathrm{T}}z \tag{5.30}$$

It holds that $V(0) = 0$. Since P is positive definite, any deviation from $\bar{\xi} = 0$ will give a positive contribution to $V(\xi)$. Similarly, the quadratic terms in (5.30) guarantee that any deviation from $(x, y_{\mathrm{c}} - g, z) = (0, 0, 0)$ gives a positive contribution to $V(\xi)$. Therefore, $V(\xi) > 0, \forall \xi \neq 0$. Further,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}V(\xi) &= \bar{\xi}^{\mathrm{T}}P\dot{\bar{\xi}} + \dot{\bar{\xi}}^{\mathrm{T}}P\bar{\xi} + x\dot{x} \\ &\quad + \alpha\beta(y_{\mathrm{c}} - g)^{\mathrm{T}}\frac{\mathrm{d}}{\mathrm{d}t}(y_{\mathrm{c}} - g) + \dot{z}^{\mathrm{T}}z \\ &= \bar{\xi}^{\mathrm{T}}(P\bar{A} + \bar{A}^{\mathrm{T}}P)\bar{\xi} - \frac{\alpha_x}{\tau_{\mathrm{a}}}x^2 + \alpha\beta(y_{\mathrm{c}} - g)^{\mathrm{T}}\frac{1}{\tau_{\mathrm{a}}}z \\ &\quad + \left(\frac{\alpha}{\tau_{\mathrm{a}}}(\beta(g - y_{\mathrm{c}}) - z) + \frac{1}{\tau_{\mathrm{a}}}f(x)\right)^{\mathrm{T}} z \\ &= -\bar{\xi}^{\mathrm{T}}Q\bar{\xi} - \frac{\alpha_x}{\tau_{\mathrm{a}}}x^2 - \frac{\alpha}{\tau_{\mathrm{a}}}z^{\mathrm{T}}z \\ &\quad + \frac{1}{\tau_{\mathrm{a}}}f(x)^{\mathrm{T}}z \leq \frac{1}{\tau_{\mathrm{a}}}f(x)^{\mathrm{T}}z, \quad \forall \xi, f(x)\end{aligned} \tag{5.31}$$

Hence, a passive mapping from $f(x)$ to z/τ_{a} can be concluded, according to the passivity theory in [Khalil, 2002] and the criteria in (5.26). Thus, the increase in stored energy is not larger than the externally added energy; see also (5.32). □

It is noteworthy that it is not straightforward to establish stability by finding a Lyapunov function [Glad and Ljung, 2000] of ξ. A common attempt is to investigate whether a weighted sum of the squared states qualifies as a Lyapunov function. However, even though $V(\xi)$ consists of such a sum, it does not qualify as a Lyapunov function. Furthermore, there is no obvious way to modify $V(\xi)$ to form a Lyapunov function.

Since $f(x)$ represents a virtual force and z/τ_{a} represents a velocity, the product of these can be interpreted as supplied power to the system. Time integration up to a time t_1 yields the following mechanical interpretation of (5.31).

$$\begin{aligned}
\underbrace{V(\xi(t_1))}_{\text{Stored energy}} &= \underbrace{V(\xi(0))}_{\text{Energy at start}} + \int_0^{t_1} \underbrace{f(x)^{\mathrm{T}} z/\tau_{\mathrm{a}}}_{\text{Supplied power}} \; \mathrm{d}t \\
&\qquad - \int_0^{t_1} \underbrace{\bar{\xi}^{\mathrm{T}} Q \bar{\xi} + \frac{\alpha_x}{\tau_{\mathrm{a}}} x^2 + \frac{\alpha}{\tau_{\mathrm{a}}} z^{\mathrm{T}} z}_{\text{Dissipation power}} \; \mathrm{d}t \\
&\leq \underbrace{V(\xi(0))}_{\text{Energy at start}} + \int_0^{t_1} \underbrace{f(x)^{\mathrm{T}} z/\tau_{\mathrm{a}}}_{\text{Supplied power}} \; \mathrm{d}t
\end{aligned} \tag{5.32}$$

Here, integrating the supplied power with respect to time yields total work done by $f(x)$. Hence, the increase in stored energy is not larger than the externally added energy, which indicates passivity.

The global exponential stability established in Sec. 5.6 does not rely on the passivity shown in this appendix. However, this appendix may contribute with some intuition of the role of $f(x)$, and of why finding a Lyapunov function of ξ is not trivial.

6

Temporally Coupled Dynamical Movement Primitives in 3D Orientation Space

6.1 Introduction

In this chapter, we extend the results in Chapter 5 to support orientation in Cartesian space. Higher levels of robot control typically operate in Cartesian space, for instance to control the pose of a robot end-effector or an unmanned aerial vehicle. However, control of orientation in Cartesian space is fundamentally limited: The rotation group SO(3) is not contractible (defined in Sec. 6.4), and only globally contractible state spaces support continuous and globally asymptotically stable feedback control systems. In this chapter, a control system for temporally coupled dynamical movement primitives (DMPs) in Cartesian space is designed. This is an extension of the control law in Chapter 5 to support 3D orientation, and equivalently an extension of [Ude et al., 2014] to include temporal coupling. Unit quaternions are used to represent orientations, and it is shown that the unit quaternion set minus one point is contractible. Finally, the efficacy of the control system is verified experimentally on an industrial robot.

6.2 Previous Research

The fundamentals of DMPs have been described in [Ijspeert et al., 2013], and earlier versions have been introduced in [Schaal et al., 2000; Ijspeert et al., 2002; Ijspeert et al., 2003]. The concept of temporal coupling for DMPs was introduced in [Ijspeert et al., 2013], and extended in Chapter 5. However, these previous DMP formulations are applicable only if the robot state space is Euclidean. As a consequence, previous applications and extensions of DMPs typically rely on Euclidean state spaces; see, *e.g.*, [Prada et al., 2014; Karlsson et al., 2017b; Talignani Landi et al., 2018a; Papageorgiou et al., 2018; Yang et al., 2018]. In [Perk and Slotine, 2006; Wensing and Slotine, 2017], a helicopter and a legged robot were controlled using DMPs, and global exponential stability was shown assuming Euclidean state spaces.

The assumption of a Euclidean state space is correct for position in Cartesian space and joint space, but not for orientation in Cartesian space. This circumstance motivated the DMP formulation proposed in [Ude et al., 2014], which supports Cartesian space including orientations. It was applied for robot programming by demonstration in [Nemec et al., 2018].

This chapter extends [Ude et al., 2014] by including temporal coupling and addressing the stability properties of the proposed control algorithm.

6.3 Problem Formulation

In this chapter, we address the question of whether the control algorithm in Chapter 5 could be extended also to incorporate orientations. Because a contractible state space is necessary for design and analysis of a continuous globally asymptotically stable control law (see Sec. 6.4), we first investigate the contractibility properties of the quaternion set used to represent orientations. Thereafter, we propose a control algorithm for temporally coupled DMPs in Cartesian space including orientations. Finally, we address the question of whether the proposed algorithm is exponentially stable. This is addressed mathematically as well as experimentally.

6.4 The Unit Quaternion Set Minus One Single Point is Contractible

Global exponential stability for temporally coupled DMPs was shown in Chapter 5, but under the assumption that the DMP state space was contractible. This is true for the spaces of joint positions as well as Cartesian positions, because these are Euclidean spaces. However, global contractibility can not directly be assumed for orientations in Cartesian space. As noted in [Mayhew et al., 2011], the rotation group SO(3) is not contractible, and therefore it is not possible for any continuous state-feedback control law to yield a globally asymptotically stable equilibrium point in SO(3) [Koditschek, 1988; Bhat and Bernstein, 2000]. A space is contractible if and only if it is homotopy equivalent to a one-point space [Crossley, 2006], which intuitively means that the space can be deformed continuously to a single point; see, *e.g.*, [Crossley, 2006] for a definition of homotopy equivalence. Contractibility is necessary for applying the contraction theory of [Lohmiller and Slotine, 1998], as done in Chapter 5. In this chapter, unit quaternions are used to parameterize SO(3). Similarly to SO(3), the unit quaternion set, $\mathbb{H}$, is not contractible. In this section, however, it is shown that it is sufficient to remove one point from $\mathbb{H}$ to yield a contractible space. Table 6.1 lists some of the notation used in this chapter.

Preliminary Topology

The fundamentals of mathematical topology and set theory are described in, *e.g.*, [Hatcher, 2002; Levy, 2002; Crossley, 2006; Schwarz, 2013]. We will use the fact that homeomorphism [Crossley, 2006] is a stronger relation than homotopy equivalence.

Lemma 6.1
If two spaces $\mathbb{X}$ and $\mathbb{Y}$ are homeomorphic, then they are homotopy equivalent. □

Proof. See Lemma 6.11 in [Crossley, 2006]. □

Lemma 6.2
Assume that $\mathbb{X} \cong \mathbb{Y}$, with a homeomorphism $f_1 : \mathbb{X} \to \mathbb{Y}$. Then $\mathbb{X}$ minus a point $p \in \mathbb{X}$, denoted $\mathbb{X} \setminus p$, is homeomorphic to $\mathbb{Y} \setminus f_1(p)$. □

Proof. Consider the function $f_2 : \mathbb{X} \setminus p \to \mathbb{Y} \setminus f_1(p)$, and let $f_2(x) = f_1(x)\ \forall x \in \mathbb{X} \setminus p$. It can be seen that f_2 is a restriction of f_1. Since a restriction of a homeomorphism is also a homeomorphism [Lehner, 1964], f_2 is a homeomorphism, and hence $\mathbb{X} \setminus p \cong \mathbb{Y} \setminus f_1(p)$. □

We will also use that homeomorphism preserves contractibility.

Lemma 6.3
If $\mathbb{X} \cong \mathbb{Y}$, and $\mathbb{X}$ is contractible, then $\mathbb{Y}$ is also contractible. □

Proof. Since $\mathbb{X} \cong \mathbb{Y}$, they are homotopy equivalent according to Lemma 6.1. In turn, $\mathbb{X}$ is contractible and therefore homotopy equivalent to a one-point space. Hence, $\mathbb{Y}$ is also homotopy equivalent to a one-point space, and therefore contractible. □

The Largest Contractible Subset of the Unit Quaternion Set

First, it will be shown that the unit sphere $\mathbb{S}^n$ (see Definition 6.1) minus a point is contractible. This will then be applied to $\mathbb{H}$, which is homeomorphic to $\mathbb{S}^3$ [LaValle, 2006].

Table 6.1 Notation used in this chapter. All quaternions represent orientations and are therefore unit quaternions, *i.e.*, quaternions of norm one. For a unit quaternion, its inverse is equal to its conjugate.

Notation		Description
$\mathbb{H}$		Unit quaternion set
$\mathbb{S}^n$	$\in \mathbb{R}^{n+1}$	Unit sphere of dimension n
y	$\in \mathbb{R}^3$	Actual robot position
g	$\in \mathbb{R}^3$	Goal position
y_c	$\in \mathbb{R}^3$	Coupled robot position
q_a	$\in \mathbb{H}$	Actual robot orientation
q_g	$\in \mathbb{H}$	Goal orientation
q_c	$\in \mathbb{H}$	Coupled robot orientation
q_0	$\in \mathbb{H}$	Initial robot orientation
ω_a	$\in \mathbb{R}^3$	Actual angular velocity
ω_c	$\in \mathbb{R}^3$	Coupled angular velocity
z, ω_z	$\in \mathbb{R}^3$	DMP states
$\mathbb{h}$		Orientation difference space
$d_{\mathrm{c}g}$	$\in \mathbb{h}$	Difference between q_c and q_g
$\alpha, \beta, k_\mathrm{v}, k_\mathrm{p}$	$\in \mathbb{R}^+$	Constant control gains
τ	$\in \mathbb{R}^+$	Time parameter
τ_a	$\in \mathbb{R}^+$	Adaptive time parameter
x	$\in \mathbb{R}^+$	Phase variable
$\alpha_x, \alpha_e, k_\mathrm{c}$	$\in \mathbb{R}^+$	Positive constants
$f(x)$	$\in \mathbb{R}^6$	Learnable virtual forcing term
$f_\mathrm{p}(x), f_\mathrm{o}(x)$	$\in \mathbb{R}^3$	Position and orientation components
N_b	$\in \mathbb{Z}^+$	Number of basis functions
m	$\in \mathbb{Z}^+$	Dimension of Cartesian configuration
$\Psi_j(x)$	$\in \mathbb{R}^6$	The j:th basis function vector
w_j	$\in \mathbb{R}^6$	The j:th weight vector
e	$\in \mathbb{R}^3 \times \mathbb{h}$	Low-pass filtered pose error
e_p	$\in \mathbb{R}^3$	Position component of e
e_o	$\in \mathbb{h}$	Orientation component of e
$\ddot{y}_\mathrm{r}, \dot{\omega}_\mathrm{r}$	$\in \mathbb{R}^3$	Reference robot acceleration
ξ	$\in \mathbb{R}^{22} \times \mathbb{h}^3$	DMP state vector
$\|\xi_i\|$	$\in \mathbb{R}^+$	2-norm of DMP state i
$\bar{q}$	$\in \mathbb{H}$	Inverse of quaternion q (same as conjugate)
h	$\in \mathbb{R}^+$	Sample period (4 ms in experiments)
$\simeq$		Homotopy equivalence
$\cong$		Homeomorphic relation

Definition 6.1
Let n be a non-negative integer. The unit sphere with dimension n is defined as

$$\mathbb{S}^n = \{p \in \mathbb{R}^{n+1} \quad | \quad \|p\|_2 = 1\} \tag{6.1}$$

□

Theorem 6.1
Let n be a non-negative integer. The unit sphere $\mathbb{S}^n$ minus a point $p \in \mathbb{S}^n$, denoted $\mathbb{S}^n \setminus p$, is contractible. □

Proof. Consider first the case $n \geq 1$. There exists a mapping from $\mathbb{S}^n \setminus p$ to $\mathbb{R}^n$ called stereographic projection from p, which is a homeomorphism. Thus, $\mathbb{S}^n \setminus p \cong \mathbb{R}^n$ [Huggett and Jordan, 2009; Schwarz, 2013]. See Fig. 6.1 for a visualization of these spaces. Since $\mathbb{R}^n$ is a Euclidean space it is contractible, and it follows from Lemma 6.3 that $\mathbb{S}^n \setminus p$ is also contractible.

Consider now the case $n = 0$. The sphere $\mathbb{S}^0$ consists of the pair of points $\{-1, 1\}$ according to Definition 6.1. Thus $\mathbb{S}^0 \setminus p$ consists of one point only, and homotopy equivalence with a one-point space is trivial. Hence $\mathbb{S}^0 \setminus p$ is contractible. □

Remark 6.1
Albeit we consider unit spheres in this chapter, it is not necessary to assume radius 1 in Theorem 6.1. Further, it is arbitrary which point $p \in \mathbb{S}^n$ to remove. □

The main result of this section is concluded by the following theorem.

Theorem 6.2
The set of unit quaternions $\mathbb{H}$ minus a point $\tilde{q} \in \mathbb{H}$, denoted $\mathbb{H} \setminus \tilde{q}$, is contractible. □

Proof. The set $\mathbb{H}$ is homeomorphic to $\mathbb{S}^3$ [LaValle, 2006]. Therefore $\mathbb{H} \setminus \tilde{q} \cong \mathbb{S}^3 \setminus p$ for some point $p \in \mathbb{S}^3$, according to Lemma 6.2. Theorem 6.1 with $n = 3$ yields that $\mathbb{S}^3 \setminus p$ is contractible, and because of the homeomorphic relation, Lemma 6.3 yields that $\mathbb{H} \setminus \tilde{q}$ is also contractible. □

It is noteworthy that the contractible subset $\mathbb{H} \setminus \tilde{q}$ is the largest possible subset of $\mathbb{H}$, because one point is the smallest possible subset to remove. Hence, it is guaranteed that no unnecessary restriction is made in Theorem 6.2, though there are other, more limited, subsets of $\mathbb{H}$ that are also contractible. Sometimes only half of $\mathbb{H}$, for instance the upper half of the quaternion hypersphere, is used to represent orientations. However, instead of continuous transitions between the half spheres this results in discontinuities within the upper half sphere [LaValle, 2006]. In the context of DMPs and automatic control, such discontinuities would cause severe obstructions,

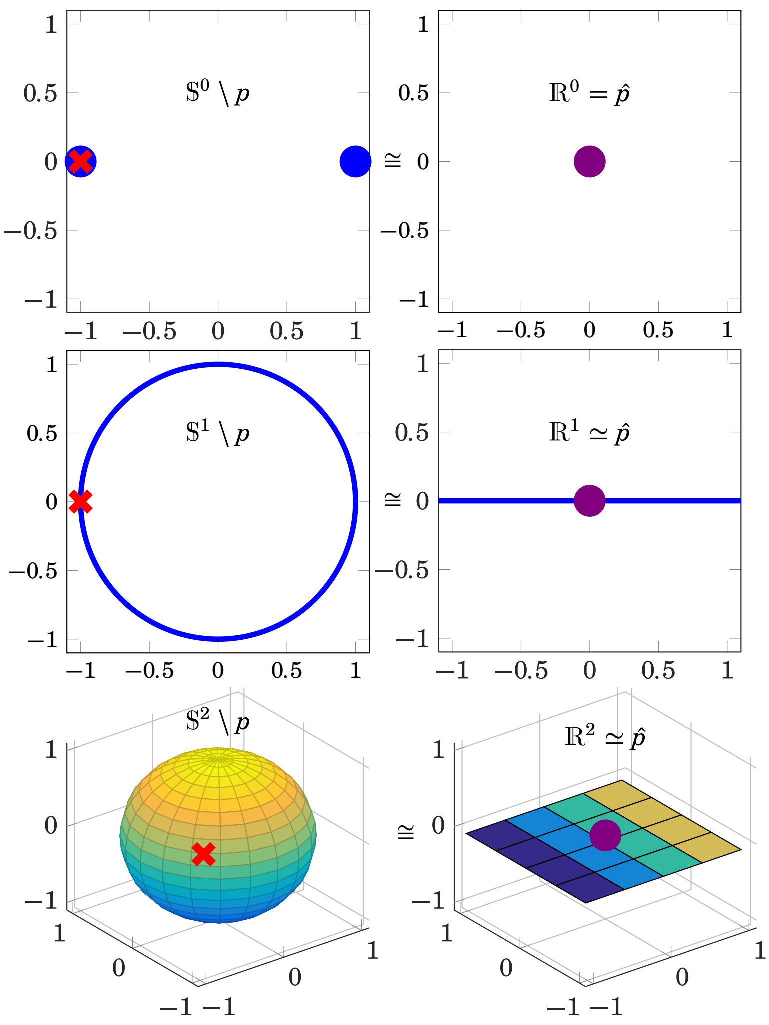

$\mathbb{S}^0 \setminus p$
$\mathbb{R}^0 = \hat{p}$
$\mathbb{S}^1 \setminus p$
$\mathbb{R}^1 \simeq \hat{p}$
$\mathbb{S}^2 \setminus p$
$\mathbb{R}^2 \simeq \hat{p}$
$\cong$
$\cong$
$\cong$

Figure 6.1 Visualization of $\mathbb{S}^n \setminus p$ (left) and $\mathbb{R}^n$ (right) for $n = 0, 1, 2$. The red cross marks a point p removed from the unit sphere. Each space to the left is homeomorphic to the corresponding space to the right, *i.e.*, $\mathbb{S}^n \setminus p \cong \mathbb{R}^n$. In turn, $\mathbb{R}^n$ is homotopy equivalent to a point (for instance $\hat{p}$ marked by a violet dot in each plot to the right) and therefore $\mathbb{S}^n \setminus p$ is contractible according to Lemma 6.3. Higher dimensions are difficult to visualize, and therefore $\mathbb{S}^2$ is commonly used to visualize parts of the quaternion set, as done in Fig. 6.9.

which motivates the search for the largest possible contractible subset of $\mathbb{H}$. One of the experiments (Setup 6.3 in Sec. 6.6) provides an example of when both half spheres are necessary for a continuous representation of the robot orientation.

6.5 Method

In this section, we augment the controller in Chapter 5 to incorporate orientation in Cartesian space. The resulting algorithm can also be seen as a temporally coupled version of the Cartesian DMPs proposed in [Ude et al., 2014]. In this chapter, we consider the control of one end-effector to limit the notation, but it would be straightforward to include several end-effectors and robots in the proposed control algorithm. The pose in Cartesian space consists of position and orientation. The position control in this chapter is the same as described in Chapter 5, except that it is also affected by the orientation through the shared time parameter τ_a in this chapter.

Similar to the approaches in [Ude, 1999; Ude et al., 2014], we define a difference between two quaternions, q_1 and q_2, as

$$d(q_1\bar{q}_2) = 2 \cdot \text{Im}[\log(q_1\bar{q}_2)] \in \mathbb{h} \tag{6.2}$$

where $\mathbb{h}$ is the orientation difference space, defined as the image of d, and Im denotes the operator that extracts the imaginary quaternion part, assuming for now that $q_1\bar{q}_2 \neq (-1, 0, 0, 0)$. This is elaborated on in Sec. 6.8. Further, we will use a shorter notation, so that for instance

$$d_{cg} = d(q_c\bar{q}_g) = 2 \cdot \text{Im}[\log(q_c\bar{q}_g)] \tag{6.3}$$

represents the difference between coupled and goal orientations. This mapping preserves the contractibility concluded in Sec. 6.4, as established by the following theorem.

THEOREM 6.3
The orientation difference space $\mathbb{h}$ is contractible. □

Proof. The mapping

$$d \quad : \quad \mathbb{H} \setminus (-1, 0, 0, 0) \to \mathbb{h} \tag{6.4}$$

has the properties necessary to qualify as a homeomorphism. It is one-to-one [Ude, 1999] and onto, continuous, and its inverse (division by 2 followed by the exponential map) is also continuous.

Further, its domain $\mathbb{H} \setminus (-1, 0, 0, 0)$ is contractible (see Theorem 6.2), and therefore its image $\mathbb{h}$ is contractible (see Lemma 6.3). □

Using the function d, a coupled DMP pose trajectory is modeled by the dynamical system

$$\tau_\mathrm{a}\dot{z} = \alpha(\beta(g - y_\mathrm{c}) - z) + f_\mathrm{p}(x) \tag{6.5a}$$

$$\tau_\mathrm{a}\dot{y}_\mathrm{c} = z \tag{6.5b}$$

$$\tau_\mathrm{a}\dot{\omega}_z = \alpha(\beta(-d_{cg}) - \omega_z) + f_\mathrm{o}(x) \tag{6.5c}$$

$$\tau_\mathrm{a}\omega_\mathrm{c} = \omega_z \tag{6.5d}$$

Here, x is a phase variable that evolves as

$$\tau_\mathrm{a}\dot{x} = -\,\alpha_x x \tag{6.6}$$

Further, $f_\mathrm{o}(x)$ is a virtual forcing term in the orientation domain, and each element of $f_\mathrm{o}(x)$, denoted $f_\mathrm{o}^i(x)$, is given by

$$f_\mathrm{o}^i(x) = \frac{\sum_{j=1}^{N_\mathrm{b}} \Psi_{i,j}(x) w_{i,j}}{\sum_{j=1}^{N_\mathrm{b}} \Psi_{i,j}(x)} x \cdot d_i(q_g \bar{q}_0) \tag{6.7}$$

where each basis function, $\Psi_{i,j}(x)$, is determined as

$$\Psi_{i,j}(x) = \exp\left(-\frac{1}{2\sigma_{i,j}^2}(x - c_{i,j})^2\right) \tag{6.8}$$

Here, σ and c denote the width and center of each basis function, respectively. The forcing term $f_\mathrm{p}(x)$ is determined accordingly, see Chapter 3. Further, the parameters of $f(x)$ can be determined based on a demonstrated trajectory by means of locally weighted regression [Atkeson et al., 1997], as described in [Ijspeert et al., 2013] and Chapter 3.

All dimensions of the robot pose are temporally coupled through the shared adaptive time parameter τ_a. Denote by y the actual position of the robot, and by q_a the actual orientation. The adaptive time parameter τ_a is determined based on the low-pass filtered difference between the actual and coupled poses as follows.

$$\dot{e}_\mathrm{p} = \alpha_e(y - y_\mathrm{c} - e_\mathrm{p}) \tag{6.9a}$$

$$\dot{e}_\mathrm{o} = \alpha_e(d_\mathrm{ac} - e_\mathrm{o}) \tag{6.9b}$$

$$e = [e_\mathrm{p}^\mathrm{T} \;\; e_\mathrm{o}^\mathrm{T}]^\mathrm{T} \tag{6.9c}$$

$$\tau_\mathrm{a} = \tau(1 + k_\mathrm{c} e^\mathrm{T} e) \tag{6.9d}$$

In (6.9c), the contributions from the position and orientation errors are weighted equally to limit the notation, though this is not necessary in an implementation. Further, we assume that the length unit for the position error has been compensated for, so that e is a dimensionless quantity. Moreover, the controller below is used to drive y to y_c, and q_a to q_c.

$$\ddot{y}_\mathrm{r} = k_\mathrm{p}(y_\mathrm{c} - y) + k_\mathrm{v}(\dot{y}_\mathrm{c} - \dot{y}) + \ddot{y}_\mathrm{c} \tag{6.10a}$$

$$\dot{\omega}_\mathrm{r} = -k_\mathrm{p} d_\mathrm{ac} - k_\mathrm{v}(\omega_\mathrm{a} - \omega_\mathrm{c}) + \dot{\omega}_\mathrm{c} \tag{6.10b}$$

This can be seen as a pose PD controller (see Sec. 6.8) together with the feedforward terms $\ddot{y}_\mathrm{c}$ and $\dot{\omega}_\mathrm{c}$. Here, $\ddot{y}_\mathrm{r}$ and $\dot{\omega}_\mathrm{r}$ denote reference accelerations sent to the internal controller of the robot, after conversion to joint values using the robot Jacobian [Spong et al., 2006]. We let $k_\mathrm{p} = k_\mathrm{v}^2/4$, so that (6.10) represents a critically damped control loop (see Chapter 5). Similarly, $\beta = \alpha/4$ (see Chapter 3 and [Ijspeert et al., 2013]). The control system is schematically visualized in Fig. 6.2. We model the 'Robot' block as a double integrator, so that $\ddot{y} = \ddot{y}_\mathrm{r}$ and $\dot{\omega}_\mathrm{a} = \dot{\omega}_\mathrm{r}$, as justified in Chapter 5 for accelerations with moderate magnitudes and time derivatives. In summary, the proposed control system is given by

$$\ddot{y} = -k_\mathrm{p}(y - y_\mathrm{c}) - k_\mathrm{v}(\dot{y} - \dot{y}_\mathrm{c}) + \ddot{y}_\mathrm{c} \tag{6.11a}$$

$$\dot{\omega}_\mathrm{a} = -k_\mathrm{p} d_\mathrm{ac} - k_\mathrm{v}(\omega_\mathrm{a} - \omega_\mathrm{c}) + \dot{\omega}_\mathrm{c} \tag{6.11b}$$

$$\dot{e} = \alpha_e \left(\begin{bmatrix} [y - y_\mathrm{c}]^\mathrm{T} & d_\mathrm{ac}^\mathrm{T} \end{bmatrix}^\mathrm{T} - e \right) \tag{6.11c}$$

$$\tau_\mathrm{a} = \tau(1 + k_\mathrm{c} e^\mathrm{T} e) \tag{6.11d}$$

$$\tau_\mathrm{a} \dot{x} = -\alpha_x x \tag{6.11e}$$

$$\tau_\mathrm{a} \dot{y}_\mathrm{c} = z \tag{6.11f}$$

$$\tau_\mathrm{a} \dot{z} = \alpha(\beta(g - y_\mathrm{c}) - z) + f_\mathrm{p}(x) \tag{6.11g}$$

$$\tau_\mathrm{a} \omega_\mathrm{c} = \omega_z \tag{6.11h}$$

$$\tau_\mathrm{a} \dot{\omega}_z = \alpha(\beta(-d_{\mathrm{c}g}) - \omega_z) + f_\mathrm{o}(x) \tag{6.11i}$$

We introduce a state vector ξ as

$$\xi = \begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ d_\mathrm{ac} \\ \omega_\mathrm{a} - \omega_\mathrm{c} \\ e \\ x \\ y_\mathrm{c} - g \\ z \\ d_{\mathrm{c}g} \\ w_z \end{pmatrix} \in \mathbb{R}^{22} \times \mathbb{h}^3 \tag{6.12}$$

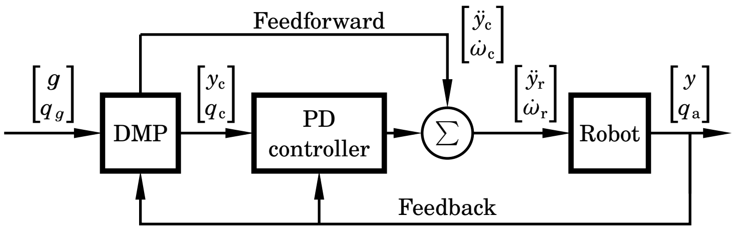

Feedforward
$\begin{bmatrix}\ddot{y}_c\\ \dot{\omega}_c\end{bmatrix}$
$\begin{bmatrix}g\\ q_g\end{bmatrix}$
DMP
$\begin{bmatrix}y_c\\ q_c\end{bmatrix}$
PD controller
$\sum$
$\begin{bmatrix}\ddot{y}_r\\ \dot{\omega}_r\end{bmatrix}$
Robot
$\begin{bmatrix}y\\ q_a\end{bmatrix}$
Feedback

Figure 6.2 The control structure for temporally coupled Cartesian DMPs. The block denoted 'Robot' includes the internal controller of the robot, together with transformations between Cartesian and joint space for low-level control. The 'DMP' block corresponds to the computations in (6.5) to (6.9). The PD controller and the feedforward terms are specified in (6.10). This forms a cascade controller, with the DMP as outer controller and the PD as the inner.

Since $\mathbb{R}^{22} \times \mathbb{h}^3$ is a product of contractible spaces (see Theorem 6.3), and such a product is itself contractible [Kahn, 1975], the state space is contractible. In Sec. 6.A, it is shown that an approximation of the proposed control system in discrete time is exponentially stable.

6.6 Experiments

The control law in Sec. 6.5 was implemented in the Julia programming language [Bezanson et al., 2014], to control an ABB YuMi [ABB Robotics, 2018] robot. The Julia program communicated with the internal robot controller through a research-interface version [Bagge Carlson and Haage, 2017] of Externally Guided Motion (EGM) [ABB Robotics, 2019c] at a sampling rate of 250 Hz.

Three different setups were used to investigate the behavior of the controller. As preparation for each setup, a temporally coupled Cartesian DMP had been determined from a demonstration by means of lead-through programming, which was available in the YuMi product by default. In each trial, the temporally coupled DMP was executed while the magnitudes of the states in (6.12) were logged.

Perturbations were introduced by physical contact with a human. This was enabled by estimating joint torques induced by the contact, and mapping these to Cartesian contact forces and torques using the robot Jacobian. A corresponding acceleration was then added to the reference acceleration as a load disturbance. However, we emphasize that this chapter is not focused on how to generate the perturbations themselves. Instead, that functionality was used only as an example of unforeseen deviations, and to

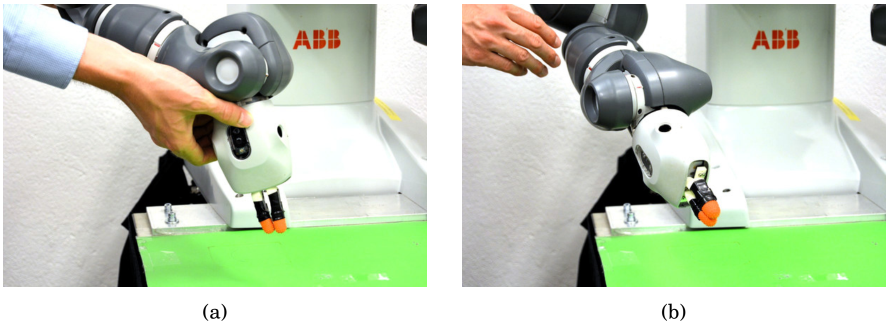


Figure 6.3 Photographs of a trial of Setup 6.1. The robot was initially released from the pose in (a), with an offset to the goal pose. In (b), the goal pose was reached.

investigate the stability properties of the proposed control algorithm.

A video of the experimental arrangement is available in [Karlsson, 2019]. The setups were as follows.

Setup 6.1 This setup is visualized in Fig. 6.3. Prior to the experiment, a test DMP that did not perform any particular task was executed, and the robot then converged to the goal pose, *i.e.*, to $y = y_c = g$ and $d_{ac} = d_{cg} = 0$. Thereafter, the operator pushed the end-effector, so that the actual pose deviated from the coupled and goal poses. The experiment was initialized when the operator released the robot arm. The purpose of this procedure was to examine the stability of the subsystem in (6.11a) to (6.11c). A total of 100 perturbations were conducted.

Setup 6.2 See Fig. 6.4. The task of the robot was to reach a work object (in this case a gore-tex graft used in cardiac and vascular surgery) from its home position. A DMP defined for this purpose was executed, and the operator introduced two perturbations during the robot movement. The purpose of this setup was to investigate the stability of the entire control system in (6.11). A total of 10 trials were conducted.

Setup 6.3 See Fig. 6.5. The task of the robot was to hand over the work object from its right arm to its left. The movement was specifically designed to require an end-effector rotation angle of more than π, thus requiring both the upper and the lower halves of the quaternion hypersphere (see Fig. 6.9), and not only one of the halves which is sometimes used [LaValle, 2006]. Such movements motivate the search for the largest possible contractible subset of $\mathbb{H}$ in Sec. 6.4. Similar to Setup 6.2, the purpose was to investigate the stability of (6.11), and 10 trials were conducted.

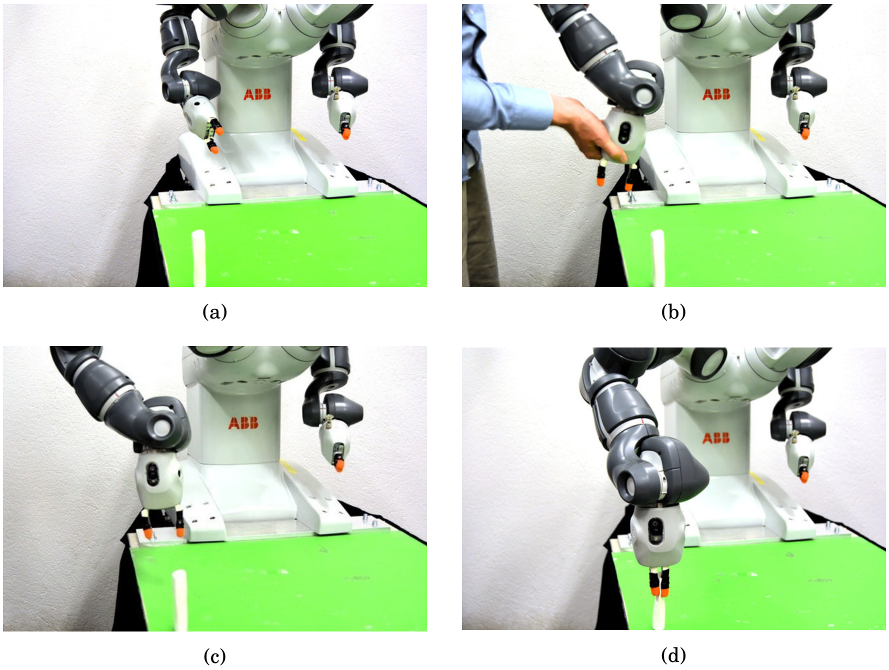


Figure 6.4 Photographs of a trial of Setup 6.2. The DMP was executed from the home position (a), and was perturbed twice on its way toward the goal (b). It recovered from these perturbations (c), and reached the goal at the work object (d).

6.7 Results

Figures 6.6 to 6.9 display data from the experiments. Figure 6.6 shows the magnitude of the states during a trial of Setup 6.1, and it can be seen that each state converged to 0 after the robot had been released. This is consistent with Theorem 6.4 in Sec. 6.A. Similarly, Figs. 6.7 and 6.8 show data from Setups 6.2 and 6.3 respectively, and it can be seen that the robot recovered from each of the perturbations. Further, each state subsequently converged to 0. This is consistent with Theorem 6.5 in Sec. 6.A. All trials in a given setup gave similar results. Further, these results suggest that the control system (6.11) is exponentially stable.

Figure 6.9 shows orientation data from Setup 6.2 (left) and Setup 6.3 (right). The upper plots show quaternions for the demonstrated paths, q_{d}, determined using lead-through programming prior to the experimental trials, relative to the goal quaternions q_g. The middle plots show coupled orientations q_{c} relative to q_g. It can be seen that the paths of q_{d} and q_{c} were similar for each of the setups, which was expected given a sufficient

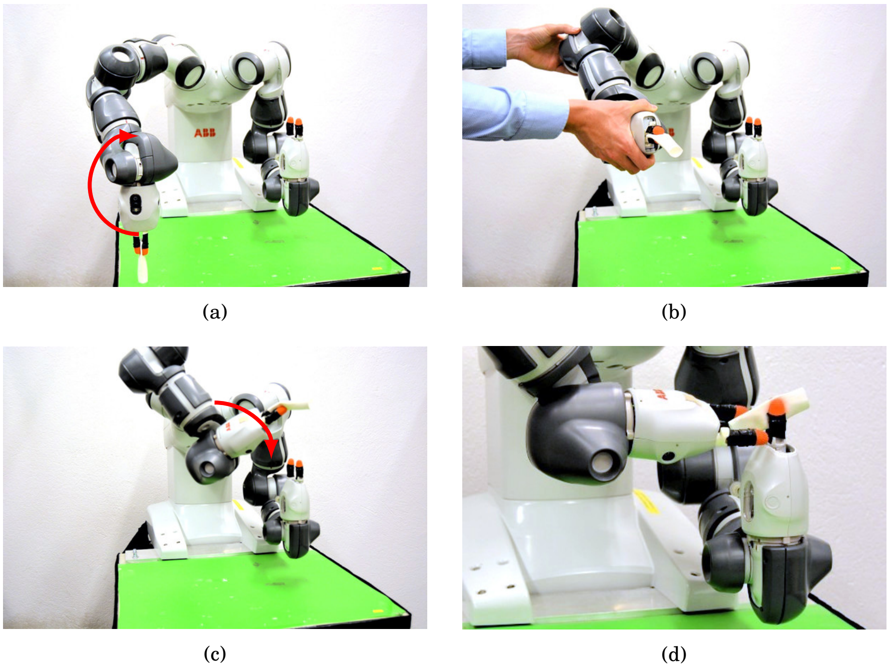


Figure 6.5 Photographs of a trial of Setup 6.3. The robot started its movement from the configuration in (a). The end-effector was rotated as indicated by the red arrows, which resulted in a rotation larger than π from start to goal. The robot was perturbed twice by the operator (b), recovered and continued its movement (c), and accomplished the handover (d).

number of DMP basis functions. The perturbations can be seen in the bottom plots, which show q_a relative to q_c. Though $q_\mathrm{a}\bar{q}_\mathrm{c}$ was very close to the identity quaternion for most of the time, it deviated significantly twice per trial as a result of the perturbations. Setup 6.3 is an example of a movement where it would not be possible to restrict the quaternions to the upper half sphere, without introducing discontinuities. This is shown in Figure 6.9, as quaternions were present not only on the upper half sphere, but also on the lower, for Setup 6.3.

6.8 Discussion

In each of the experiments, the robot recovered from the perturbations and subsequently reached the goal pose, which was the desired behavior. Because each state converged to 0, we conclude that the experimental results verify Theorem 6.4 and Theorem 6.5; see Sec. 6.A. Further, the behavior

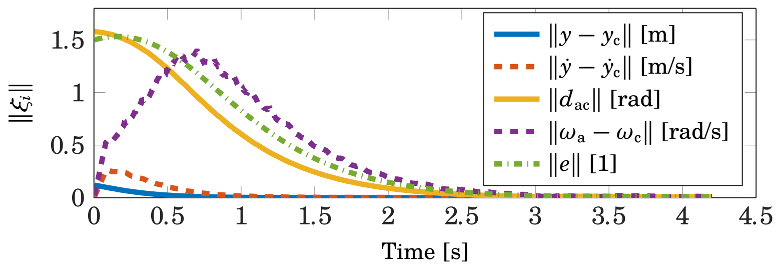


Figure 6.6 Data from a trial of Setup 6.1. The notation $\|\cdot\|$ represents the 2-norm, and the unit symbol [1] indicates dimensionless quantity. The experiment was initialized with some position error $y - y_c$ and orientation error d_{ac}. The operator released the robot at $t = 0$. It can be seen that each state converged to 0.

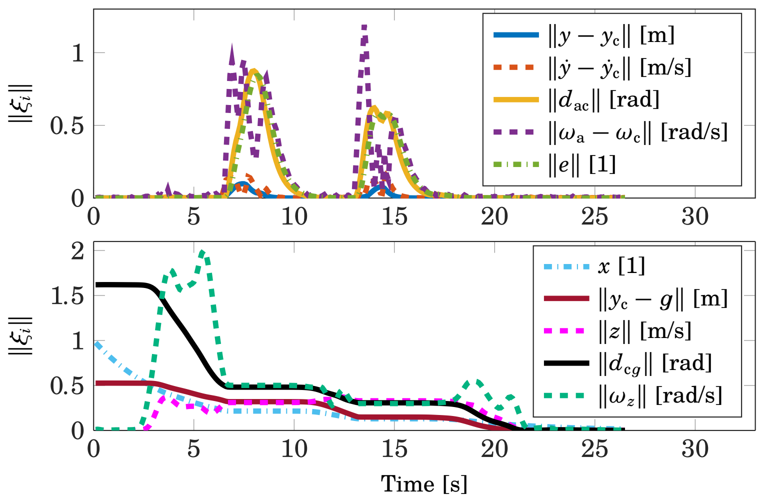


Figure 6.7 Data from a trial of Setup 6.2. Consider first the upper plot. The two perturbations are clearly visible, and these were recovered from as the states converged to 0. In the lower plot, it can be seen that the time evolution of the states was slowed down in the presence of perturbations. It can further be seen that each of the states converged to 0.

corresponds to that in Chapter 5, except that orientations in Cartesian space are now supported. Most of the discussion in Chapter 5 is therefore valid also for these results, and is not repeated here.

There are two reasons for discretizing the control system in Sec. 6.A. Firstly, it shows that the system can be integrated without violating the unit

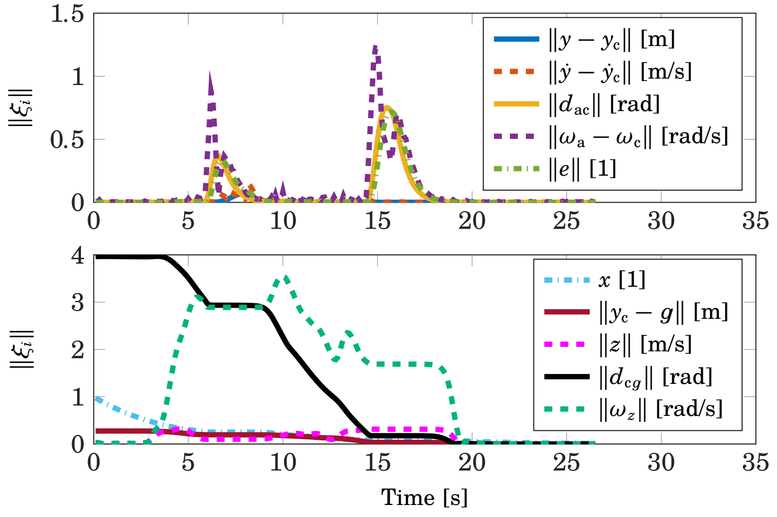


Figure 6.8 Data from a trial of Setup 6.3. The organization is the same as in Fig. 6.7, and similar conclusions can be drawn. In addition, the required rotation angle from start to goal was larger than π in this setup, which corresponds to $\|d_{cg}\|$ being larger than π initially.

length of the quaternions, which is sometimes problematic. This is possible because the quaternion difference in (6.2) can be updated by, for instance, the forward Euler method, without violating quaternion unit length. Applying such updates directly on quaternions would have resulted in loss of unit length. This is particularly important for q_c, which is not updated by measurements but by simulated integration also during DMP operation on real robots. Secondly, the discretization yields conditions on the sampling period h, which should be short enough in relation to the control gains to retain exponential stability. These conditions can easily be extracted from the eigenvalues in (6.20), (6.22) and (6.26), which should be inside the unit circle. Further, these conditions are easy to satisfy with reasonable gains and sampling periods, and they were therefore assumed to be satisfied in the proofs in this chapter.

It should be emphasized that the control system analyzed in Sec. 6.A is a simplified version of the proposed control system. The view of the feedback control in (6.10b) as PD control and the representation of the control system in Sec. 6.A rely on the approximation that the time derivative of the quaternion difference in (6.2) is equal to the angular velocity, *i.e.*, $\dot{d}(q) \approx \omega$. This is exactly true only when the rotation takes place around a fixed axis [Boyle, 2017]. For the sake of proving stability, however, the

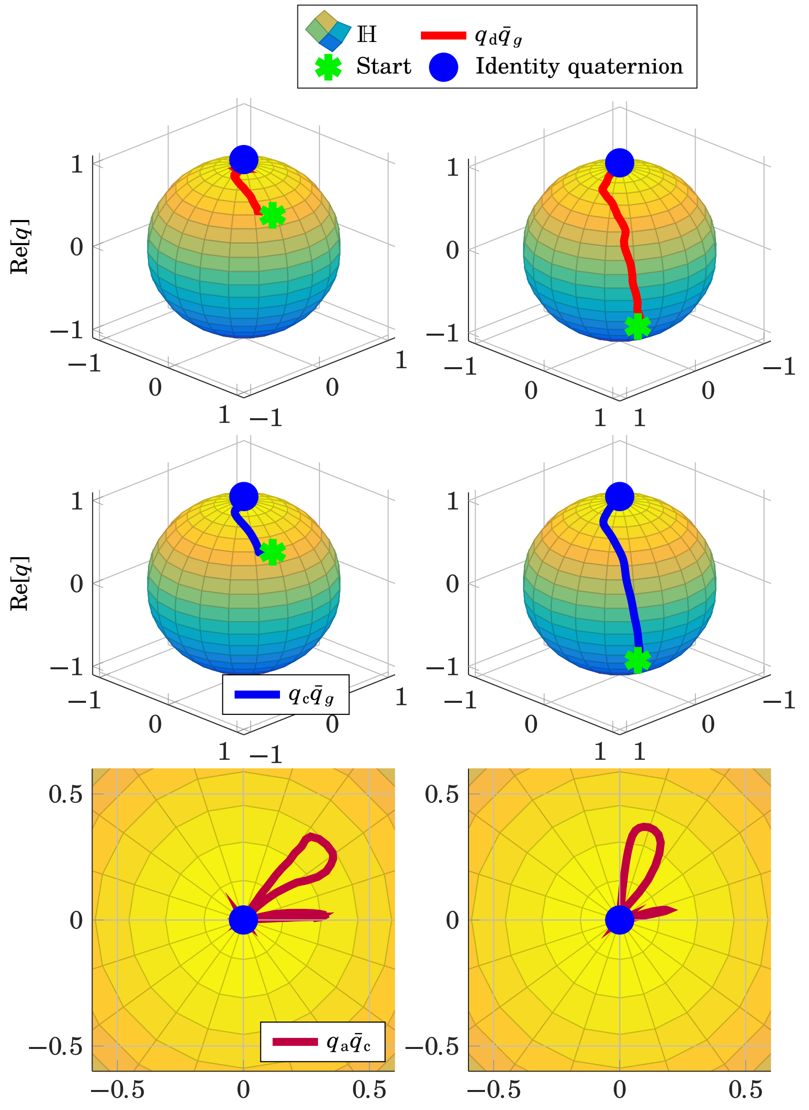


Figure 6.9 Orientation data from Setup 6.2 (left) and Setup 6.3 (right). Quaternions have been projected onto $\mathbb{S}^2$ for the purpose of visualization. Vertical axes represent quaternion real parts, and horizontal axes represent the first two imaginary elements with magnitudes adjusted to yield unit length of the resulting projection. The bottom plots show the quaternion set seen from above, and hence their real axes are directed out from the figure.

approximation was only useful for showing that q_a is driven to q_c, and q_c to q_g after the virtual forcing term in practice has vanished. Hence, detours due to the virtual forcing term, which indeed might violate the assumption of a fixed rotation axis, are not as problematic as it might seem. A formal proof that the exact proposed control system is exponentially stable would enhance the contribution of this chapter, but remains as future research. Nevertheless, it has now been shown that the topology of $\mathbb{h}$ does not prohibit a globally exponentially stable control system.

The magnitude of the difference between two quaternions, $\|d(q_1\bar{q}_2)\|$, corresponds to the length of a geodesic curve connecting q_1 and q_2 [Ude, 1999]. This results in proper scaling between orientation difference and angular velocity in the DMP control algorithm, as explained in [Ude et al., 2014]. This is the reason why the quaternion difference in (6.2) was used in [Ude et al., 2014] and in this chapter.

In Sec. 6.4, the largest possible contractible subset of $\mathbb{H}$ was found as $\mathbb{H}\setminus\tilde{q}$. Hence, it is not necessary to remove a large proportion of the quaternion set, which is sometimes done. For instance, sometimes the lower half of the quaternion hypersphere is removed [LaValle, 2006], which is unnecessarily limiting. The results from Setup 6.3 show that this proposed method works also when it is necessary to use both half spheres, see Fig. 6.9. In Sec. 6.5, the removed point $\tilde{q}$ was chosen as $(-1, 0, 0, 0)$, which corresponds to a full 2π rotation from the identity quaternion. A natural question is therefore how to handle the case where $(-1, 0, 0, 0)$ is visited by $q_\mathrm{a}\bar{q}_\mathrm{c}$ or $q_\mathrm{c}\bar{q}_g$. In theory, almost any control signal could be used to move the orientations away from this point, and in practice a single point would never be visited because it is infinitely small. However, in practice some care should be taken in a small region around $(-1, 0, 0, 0)$, because of possible numerical difficulties and rapidly changing control signals.

In this chapter, the same control gains were used in the position domain as in the orientation domain. This was done in order to limit the notation, but is not actually required.

The functionality presented in this chapter was demonstrated at the final review of the SARAFun project, see Chapter 2 and [SARAFun, 2019]. It was used for motion control in combination with the obstacle avoidance in [Stavridis et al., 2017] on the main demonstration platform, and in combination with the friction estimation in [Bagge Carlson, 2019] on the separate demonstration platform.

An interesting direction of future research is to use the proposed controller to warm-start reinforcement learning for robotic manipulation. Reinforcement learning with earlier DMP versions has been investigated in, *e.g.*, [Li et al., 2018].

6.9 Conclusion

In this chapter, it was first shown that the unit quaternion set minus one point is contractible. A contractible state space allows for continuous and exponentially stable control systems. Thereafter, a control algorithm for DMPs with temporal coupling in Cartesian space was designed. The proposed DMP functionality was verified experimentally on an industrial robot. Further, exponential stability was proven mathematically for an approximation of the proposed control system. A video that shows the experiments is available in [Karlsson, 2019].

Appendix A. Stability Analysis

In this appendix, an approximation of the proposed control system is shown to be exponentially stable. The entire control system in Sec. 6.5 is given by

$$\ddot{y} = -k_{\mathrm{p}}(y - y_{\mathrm{c}}) - k_{\mathrm{v}}(\dot{y} - \dot{y}_{\mathrm{c}}) + \ddot{y}_{\mathrm{c}} \tag{6.13a}$$

$$\dot{\omega}_{\mathrm{a}} = -k_{\mathrm{p}} d_{\mathrm{ac}} - k_{\mathrm{v}}(\omega_{\mathrm{a}} - \omega_{\mathrm{c}}) + \dot{\omega}_{\mathrm{c}} \tag{6.13b}$$

$$\dot{e} = \alpha_e \left(\begin{bmatrix} [y - y_{\mathrm{c}}]^{\mathrm{T}} & d_{\mathrm{ac}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} - e \right) \tag{6.13c}$$

$$\tau_{\mathrm{a}} = \tau(1 + k_{\mathrm{c}} e^{\mathrm{T}} e) \tag{6.13d}$$

$$\tau_{\mathrm{a}} \dot{x} = -\alpha_x x \tag{6.13e}$$

$$\tau_{\mathrm{a}} \dot{y}_{\mathrm{c}} = z \tag{6.13f}$$

$$\tau_{\mathrm{a}} \dot{z} = \alpha(\beta(g - y_{\mathrm{c}}) - z) + f_{\mathrm{p}}(x) \tag{6.13g}$$

$$\tau_{\mathrm{a}} \omega_{\mathrm{c}} = \omega_z \tag{6.13h}$$

$$\tau_{\mathrm{a}} \dot{\omega}_z = \alpha(\beta(-d_{\mathrm{c}g}) - \omega_z) + f_{\mathrm{o}}(x) \tag{6.13i}$$

The state vector is given by

$$\xi = \begin{pmatrix} y - y_{\mathrm{c}} \\ \dot{y} - \dot{y}_{\mathrm{c}} \\ d_{\mathrm{ac}} \\ \omega_{\mathrm{a}} - \omega_{\mathrm{c}} \\ e \\ x \\ y_{\mathrm{c}} - g \\ z \\ d_{\mathrm{c}g} \\ w_z \end{pmatrix} \in \mathbb{R}^{22} \times \mathbb{h}^3 \tag{6.14}$$

Because $\mathbb{R}^{22} \times \mathbb{h}^3$ is a product of contractible spaces (see Theorem 6.3), and such a product is itself contractible [Kahn, 1975], we note that the state space in (6.14) is contractible. This is a necessary property for the contraction theory applied in this appendix.

In the following, we will consider a representation of the system in discrete time, based on the approximation that the time derivative of the quaternion difference in (6.2) is equal to the angular velocity, *i.e.*, $\dot{d}(q) \approx \omega$. This is elaborated on in Sec. 6.8. Using the forward Euler method, a discrete state-space representation of the system is given by

$$\xi(t+h) = \begin{pmatrix} y - y_\mathrm{c} + h(\dot{y} - \dot{y}_\mathrm{c}) \\ -hk_\mathrm{p}(y - y_\mathrm{c}) + (1 - k_\mathrm{v}h)(\dot{y} - \dot{y}_\mathrm{c}) \\ d_\mathrm{ac} + h(\omega_\mathrm{a} - \omega_\mathrm{c}) \\ -hk_\mathrm{p}d_\mathrm{ac} + (1 - hk_\mathrm{v})(\omega_\mathrm{a} - \omega_\mathrm{c}) \\ e + h\alpha_e\left(\begin{bmatrix}[y - y_\mathrm{c}]^\mathrm{T} & d_\mathrm{ac}^\mathrm{T}\end{bmatrix}^\mathrm{T} - e\right) \\ \left(1 - h\dfrac{\alpha_x}{\tau_\mathrm{a}}\right)x \\ y_\mathrm{c} - g + \dfrac{h}{\tau_\mathrm{a}}z \\ z + \dfrac{h}{\tau_\mathrm{a}}\left[\alpha(\beta(g - y_\mathrm{c}) - z) + f_\mathrm{p}(x)\right] \\ d_{\mathrm{c}g} + \dfrac{h}{\tau_\mathrm{a}}\omega_z \\ w_z + \dfrac{h}{\tau_\mathrm{a}}\left[\alpha(\beta(-d_{\mathrm{c}g}) - \omega_z) + f_\mathrm{o}(x)\right] \end{pmatrix}(t) \tag{6.15}$$

Here, h represents the sampling period, and t is discrete time so that $t = kh$ for some integer k. Because of the adaptive time scale from τ_a, and the forcing terms $f_\mathrm{p}(x)$ and $f_\mathrm{o}(x)$, the system is nonlinear. To structure the stability analysis, we utilize that (6.15) is a hierarchical system, and analyze one subsystem at a time. A similar strategy was applied in Chapter 5, though for a purely Euclidean state space. In this context, the following relation is useful.

PROPOSITION 6.1
If $x_1(t+h) = g_1(x_1(t))$ is contracting, and $x_2(t+h) = g_2(x_1(t), x_2(t))$ is contracting for each fixed x_1, then the hierarchy

$$\begin{pmatrix} x_1(t+h) \\ x_2(t+h) \end{pmatrix} = \begin{pmatrix} g_1(x_1(t)) \\ g_2(x_1(t), x_2(t)) \end{pmatrix} \tag{6.16}$$

is contracting. □

This is a discrete version of Proposition 2 in [Wensing and Slotine, 2017] applied to autonomous systems.

In the following, we assume that h is sufficiently short in relation to the control gains, so that the discretization itself does not cause instability; see Sec. 6.8.

The first five lines in (6.15) do not depend on the rest of the system, and can be analyzed separately as the following linear subsystem. Denote by ξ_1 the state vector of the subsystem, so that

$$\xi_1(t+h) = \Phi_1 \xi_1(t) \tag{6.17}$$

where ξ_1 is given by

$$\xi_1 = \begin{pmatrix} y - y_\mathrm{c} \\ \dot{y} - \dot{y}_\mathrm{c} \\ d_\mathrm{ac} \\ \omega_\mathrm{a} - \omega_\mathrm{c} \\ e_\mathrm{p} \\ e_\mathrm{o} \end{pmatrix} \tag{6.18}$$

Further, denote by I the identity matrix and by 0 the zero matrix, of due sizes. The system matrix is given by

$$\Phi_1 = \begin{pmatrix} I & hI & 0 & 0 & 0 & 0 \\ -hk_\mathrm{p}I & (1-hk_\mathrm{v})I & 0 & 0 & 0 & 0 \\ 0 & 0 & I & hI & 0 & 0 \\ 0 & 0 & -hk_\mathrm{p}I & (1-hk_\mathrm{v})I & 0 & 0 \\ h\alpha_e I & 0 & 0 & 0 & (1-h\alpha_e)I & 0 \\ 0 & 0 & h\alpha_e I & 0 & 0 & (1-h\alpha_e)I \end{pmatrix} \tag{6.19}$$

Theorem 6.4
The dynamical system defined by (6.17) for Cartesian DMP operation is a contraction and has $\xi_1 = 0$ as fixed point. □

Proof. If $\xi_1(t) = 0$, it can be seen in (6.17) that also $\xi_1(t+h) = 0$, and $\xi_1 = 0$ is therefore a fixed point. The eigenvalues of Φ_1 are given by

$$\lambda_{1,\dots,2m} = 1 - \frac{hk_\mathrm{v}}{2} \tag{6.20a}$$

$$\lambda_{2m+1,\dots,3m} = 1 - h\alpha_e \tag{6.20b}$$

Each eigenvalue is strictly within the unit circle since $h, k_\mathrm{v}, \alpha_e > 0$, and the system is therefore globally exponentially stable. Because it is also a linear time-invariant system, contraction can be concluded. □

Next, we will show contraction for a larger subsystem of (6.15).

PROPOSITION 6.2
The system defined by

$$\xi_1(t+h) = \Phi_1 \xi_1(t) \tag{6.21a}$$

$$x(t+h) = (1 - h\frac{\alpha_x}{\tau_\mathrm{a}})x(t) \tag{6.21b}$$

is contracting and has the origin as fixed point. □

Proof. The origin is a fixed point since $(\xi_1(t), x(t)) = (0,0) \Rightarrow (\xi_1(t+h), x(t+h)) = (0,0)$. Further, it is known from Theorem 6.4 that (6.21a) is contracting with the origin as fixed point. Hence, for the fixed point of (6.21a), it holds that $\tau_\mathrm{a} = \tau$. For the fixed point of (6.21a), x therefore evolves as the linear system

$$x(t+h) = (1 - h\frac{\alpha_x}{\tau})x(t) \tag{6.22}$$

which is a contraction because the eigenvalue is strictly within the unit circle (since $h, \alpha_x, \tau > 0$). Because (6.21a) is contracting and (6.21b) is contracting for the fixed point of (6.21a), it follows from Proposition 6.1 that the hierarchy (6.21) is also contracting. □

Finally we address the stability of the whole DMP system in (6.15).

THEOREM 6.5
The system in (6.15) for Cartesian DMP operation has the origin as globally exponentially stable equilibrium point. □

Proof. The system in (6.15) is a hierarchy from (6.21) to the rest of (6.15) (through τ_a and x). It is known from Proposition 6.2 that (6.21) is contracting with the origin as fixed point. For the fixed point of (6.21), we specifically have $\tau_\mathrm{a} = \tau$ and $x = 0$, and the rest of (6.15) is linear and can be written as

$$\xi_2(t+h) = \Phi_2 \xi_2(t) \tag{6.23}$$

where

$$\xi_2 = \begin{pmatrix} y_\mathrm{c} - g \\ z \\ d_{cg} \\ w_z \end{pmatrix} \tag{6.24}$$

so that $\xi = [\xi_1^{\mathrm{T}} \;\; x \;\; \xi_2^{\mathrm{T}}]^{\mathrm{T}}$, and

$$\Phi_2 = \begin{pmatrix} I & \frac{h}{\tau} I & 0 & 0 \\ -\frac{h\alpha\beta}{\tau} I & \left(1 - \frac{h\alpha}{\tau}\right) I & 0 & 0 \\ 0 & 0 & I & \frac{h}{\tau} I \\ 0 & 0 & -\frac{h\alpha\beta}{\tau} I & \left(1 - \frac{h\alpha}{\tau} I\right) \end{pmatrix} \tag{6.25}$$

The eigenvalues of Φ_2 are given by

$$\lambda_{1,\ldots,2m} = 1 - \frac{h\alpha}{2\tau} \tag{6.26}$$

and because they are all strictly within the unit circle since $h, \alpha, \tau > 0$, (6.23) is contracting. It now follows from Proposition 6.1 that the hierarchy (6.15) is contracting. Hence, it converges globally exponentially to a single trajectory. Since one solution to (6.15) is $\xi = 0$, any trajectory must converge exponentially to this point. □

7
Detection and Control of Contact-Force Transients

7.1 Introduction

Robot programming is traditionally done using position-based control. Sequences of movements are specified with details such as velocities, target positions, *etc*. Robot positions in free space are uncomplicated to visualize and simulate, for instance with software tools such as rviz [ROS, 2018] and Gazebo [Gazebo, 2018], and successful movement can be verified by comparing measured position with target position. Control in the contact-force domain is also possible [Olsson et al., 2002; García et al., 2006], but this is a more difficult control problem and forces are only indirectly visible [Johansson et al., 2015]. Nevertheless, many applications rely on force control. In friction-stir welding, for instance, a rotating tool is pushed against work pieces with a specified contact force to generate heat from friction [Cook et al., 2004; De Backer, 2014; De Backer and Bolmsjö, 2014; Bagge Carlson et al., 2016; Karlsson et al., 2016]. In lead-through programming, motor torques are commanded to help the programmer in overcoming joint friction [Stolt et al., 2015a].

Whereas the previous chapters focused on algorithms for motion control, the aim of this chapter is to use contact forces for validation of manipulation subtasks. In robotic manipulation tasks, such as assembly tasks, it might not be possible to validate subtasks based on robot position. One reason is that the position tolerances are typically lower than the position uncertainties. The uncertainties are introduced both by positioning of the robot end-effector, which can be done within 1 mm on a typical industrial robot, and of the work objects. Further, early detection is usually desirable, and in some cases force data may indicate completion earlier than position data. Force measurements were used to validate subtasks in robotic assembly in, *e.g.*, [Stolt et al., 2011; Stolt et al., 2012b; Linderoth, 2013; Stolt, 2015].

Conditions for the validation consisted of contact-force thresholds and were programmed manually. This programming requires expert knowledge and is time consuming.

In contrast, the research in [Stolt et al., 2015b] allowed for the programmer to provide the robot with examples instead of programming explicitly. Thereby, [Stolt et al., 2015b] contributed to more accessible robot programming. Measured contact-force transients generated during robotic assembly were recognized based on a classification model acquired using machine learning, and used for validation. The final classification model consisted of a support vector machine (SVM) [Bishop, 2007; Murphy, 2012], trained by solving a convex optimization problem in CVX [Grant and Boyd, 2008; CVX Research Inc., 2019]. Some hyperparameters were determined in an initial training phase, where the classification was based partly on a linear model learned using the least-squares method [Bishop, 2007; Murphy, 2012], and partly on several learnable threshold detectors combined into a boosting classifier. This approach reduced the assembly time, compared to using force thresholds only. It also removed the necessity to determine and specify any force threshold manually, which would have required considerable engineering work and explicit robot programming. As compared to using position criteria, the approach in [Stolt et al., 2015b] also enabled robots to switch between movements at the right time, despite any position uncertainties, thus avoiding to push unnecessarily hard on work objects or to leave any task unfinished.

In this chapter, we continue the work presented in [Stolt et al., 2015b], with the following extensions. In [Stolt et al., 2015b], a force/torque sensor was used to measure the contact force/torque. Such sensors and systems are usually expensive, with costs comparable to the robot itself. If attached to the wrist of the robot, it would introduce extra weight that the robot would have to lift and move. Further, some robot models do not support any seamless attachment of such sensors. If the force sensor would be attached to an object in the work space, *e.g.*, a table, this would imply restrictions on where the assembly could take place. In this work, we therefore propose a method based on robot joint-torque measurements on the motor side for the detection, thus avoiding the requirement of an external sensor. Motor-torque measurements are typically available, at least for the manufacturers, which is a benefit. However, measuring on the motor side also introduces a new difficulty. Due to friction, backlash, and elasticity between the arm side and the motor side of each joint, some information is lost when using motor torques as compared to an external force/torque sensor. A robot joint model is visualized in Fig. 6.1 in [Olofsson, 2015]. Motor torque data could be useful also when force/torque sensors are available, since the redundant information could increase the confidence of state estimates. In this thesis, the gear ratio between the motor side and the arm side has been compensated for, so that the presented joint torques are of the same magnitude as on the arm side even though they have been measured on the motor side.

The research presented in this chapter also extends [Stolt et al., 2015b] by investigating whether a given detection model could generalize to tasks that involve new objects, from which no data have been used to determine the model. In order to investigate whether robot joint torques contain enough information to distinguish transients, we follow a machine-learning procedure. First, we gather data to determine a recurrent neural network (RNN), which is an artificial neural network specialized in processing sequential data [Graves, 2012; Goodfellow et al., 2016], and subsequently we evaluate the performance of the RNN on new data.

The procedure proposed does not rely on any assumption of what tasks or work objects are considered, as long as distinguishable joint-torque transients are generated. In this chapter, we evaluate the procedure on the manipulation scenarios shown in Figs. 7.1 and 7.2. These scenarios also serve as examples of how the procedure could be used in practice.

Machine learning for analyzing contact forces in robotic assembly was also applied in [Rodriguez et al., 2010], where force measurements were used as input to an SVM, to distinguish between successful and failed assemblies. A verification system, specialized in snap-fit assembly, was developed in [Rojas et al., 2012]. Similar to [Stolt et al., 2015b] and [Rodriguez et al., 2010], a force/torque sensor was assumed in [Rojas et al., 2012]. Such a requirement has been avoided in some previous research, by using internal robot sensors instead. For instance, a method to estimate contact forces from joint torques was presented in [Linderoth et al., 2013]. Further, force-controlled assembly without a force sensor was achieved in [Stolt et al., 2012b], by estimating contact forces from position errors in the internal controller of the robot.

7.2 Problem Formulation

Autonomous robotic manipulation requires in each subtask a validation procedure, to determine when to switch to the next subtask. Without such validation, subtask completion may not be verified, thus jeopardizing successive steps and, eventually, the successful completion of the entire task. In this research, we address the question of whether robot joint torques on the motor side could be used to recognize contact-force transients during robotic assembly, despite uncertainties introduced by, *e.g.*, joint friction, in order to automatically confirm completion of manipulation tasks. We further investigate how long parts of the transients that need to be included as input for the detection algorithm, to achieve satisfactory performance of the detection. Finally, we investigate whether a detection model trained on

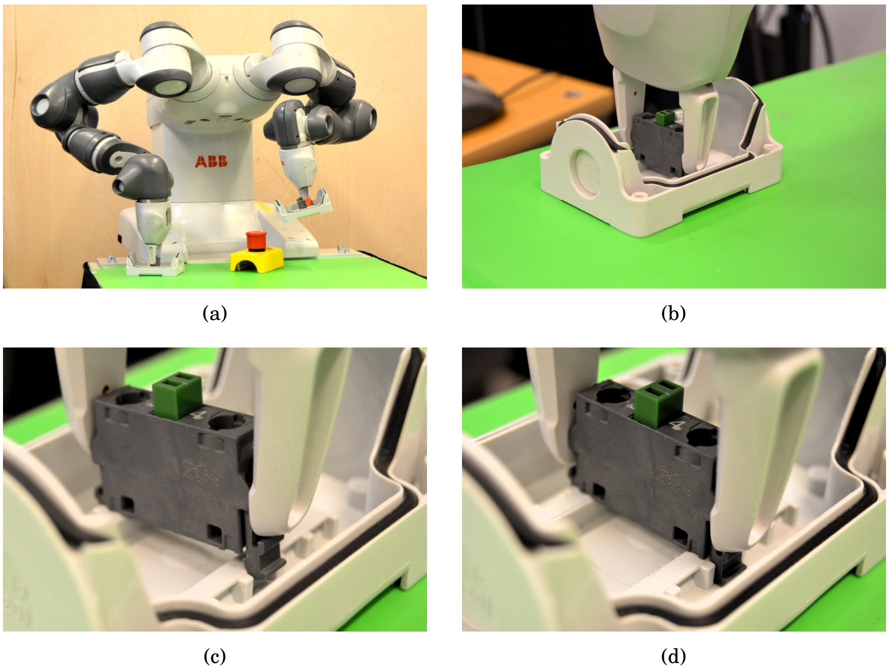


Figure 7.1 An ABB YuMi robot [ABB Robotics, 2018] was used in the experiments (a). The experimental setup for the switch assembly task is shown in (b), (c), and (d). The robot gripper, box, and switch are shown in (b). The robot grasped the switch, and attached it to the box by pushing downwards. The downward motion began in (c), where the switch was not yet snapped into place. It ended when the assembly was completed, in (d). These photos were taken during the experimental evaluation (see Sec. 7.4), and the same setup was used for gathering training and test data (see Sec. 7.3).

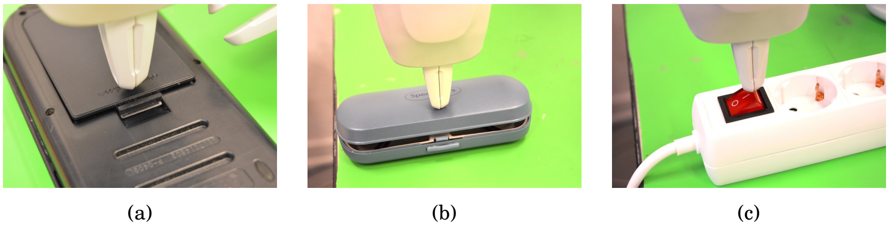


Figure 7.2 Experimental setups for evaluation of the RNN model on objects that had not been used for acquiring test or training data (Setup 7.2). The pocket calculator and its battery cover are shown in (a), the spectacle case is shown in (b), and the power switch on the extension cord is shown in (c). For each setup, the robot was programmed to move the gripper downwards until a transient was detected, and subsequently move the gripper upwards.

Table 7.1 Notation used in this chapter.

Notation		Description
T	$\in \mathbb{Z}^+$	Number of samples in torque sequence
n_{pre}	$\in \mathbb{Z}^+$	Number of samples before transient peak
n_{post}	$\in \mathbb{Z}^+$	Number of samples after transient peak
n_{ch}	$\in \mathbb{Z}^+$	Number of input channels
$\tanh(\cdot)$	$\mathbb{R}^n \rightarrow \mathbb{R}^n$	Element-wise hyperbolic tangent function
$\bar{\tau}$	$\in \mathbb{R}^{n_{\text{ch}}}$	Robot joint torque
h	$\in \mathbb{R}^{n_{\text{ch}}}$	Activation of hidden RNN neurons
U, W	$\in \mathbb{R}^{n_{\text{ch}} \times n_{\text{ch}}}$	RNN weight matrices
b	$\in \mathbb{R}^{n_{\text{ch}}}$	RNN bias vector
V	$\in \mathbb{R}^{2 \times n_{\text{ch}}}$	RNN weight matrix
c	$\in \mathbb{R}^2$	RNN bias vector
o	$\in \mathbb{R}^2$	RNN output vector
e	$\in \mathbb{R}$	Base of the natural logarithm
$\hat{p}$	$\in \mathbb{R}^2$	Estimated probability distribution
p	$\in \mathbb{R}^2$	Target label for given torque sequence
L	$\in \mathbb{R}$	Loss function

data from manipulation of a certain object could be used for detection in similar tasks but with different objects and robots.

7.3 Method

In this section, the proposed machine-learning procedure to determine a force/torque transient-detection model from training and test data is detailed. The assembly scenario used to acquire the data consisted of attaching a switch to a box, see Fig. 7.1. The objective for the robot was to move toward the box while holding the switch, thus pushing the switch against the box, until it snapped into place. The robot should detect the completion of the task automatically, stop moving toward the box, and possibly start a new movement. Table 7.1 lists some of the notation used in this chapter.

Sequence Model

An RNN [Graves, 2012; Goodfellow et al., 2016] was used as a sequence classifier. This choice is discussed in Sec. 7.6. It had a sequence of joint torques as input, one single output indicating whether the sequence contained a given transient or not, one hidden layer, and recurrent connections between its hidden neurons. Each input torque sequence consisted of $T = n_{\text{pre}} + 1 + n_{\text{post}}$ time samples, where n_{pre} and n_{post} were determined as explained later in this section. In turn, each time sample consisted of $n_{\mathrm{ch}} = 7$ channels, *i.e.*, one channel per robot joint. The dimension of the hidden layer was chosen to be the same as the number of input channels, n_{ch}.

Denote by $h^{(t)}$ the activation of the hidden neurons at time step t. The activation was defined recursively as

$$h^{(1)} = \tanh(b + U\bar{\tau}^{(1)}) \tag{7.1a}$$

$$h^{(t)} = \tanh(b + Wh^{(t-1)} + U\bar{\tau}^{(t)}) \qquad t \in [2; T] \tag{7.1b}$$

where U and W are weight matrices, both of size $n_{\mathrm{ch}} \times n_{\mathrm{ch}}$, b is a bias vector with dimension n_{ch}, and $\bar{\tau}^{(t)}$ is the input joint torque at time t. Further, $\tanh(\cdot)$ represents the hyperbolic tangent function. After reading an entire input sequence, the RNN produced one output $o^{(T)}$ given by

$$o^{(T)} = c + Vh^{(T)} \tag{7.2}$$

where V is a weight matrix of size $2 \times n_{\mathrm{ch}}$, and c is a bias vector with dimension 2. Finally, the softmax operation was applied to generate $\hat{p}$, a vector that represents the normalized probabilities of whether a transient is present.

$$\hat{p} = [\hat{p}_1 \;\; \hat{p}_2]^{\mathrm{T}} = \left[\frac{e^{o_1^{(T)}}}{e^{o_1^{(T)}} + e^{o_2^{(T)}}} \quad \frac{e^{o_2^{(T)}}}{e^{o_1^{(T)}} + e^{o_2^{(T)}}} \right]^{\mathrm{T}} \tag{7.3}$$

Here, $o_i^{(T)}$ represents the i:th element of the output vector, $\hat{p}_1$ represents the estimated probability that a transient is present, and $\hat{p}_2$ is the estimated probability that a transient is not present, so that $\hat{p}_1 + \hat{p}_2 = 1$. If $\hat{p}_1$ was larger than a desired decision boundary (0.5 was chosen, see Sec. 7.6), the data point was classified as positive, *i.e.*, it was indicated that the task was completed within the sequence. Vice versa, if the first element was less than the decision boundary, the data point was classified as negative. The RNN architecture is visualized in Fig. 7.3.

Acquisition of Training Data and Test Data

Training data and test data were obtained as follows. The right arm of the robot was used to grasp the switch, just above the box, as shown in Fig. 7.1. Thereafter, a reference velocity was sent to the internal controller of the robot, causing the robot gripper to move toward the box at 1.5 mm/s, thus pushing the switch against the box. Once the switch was snapped into place, the robot was stopped manually by the robot operator. The robot joint torques were recorded in 250 Hz. The torque transient, induced by the snap-fit, was labeled manually, and used to form a positive data point. This

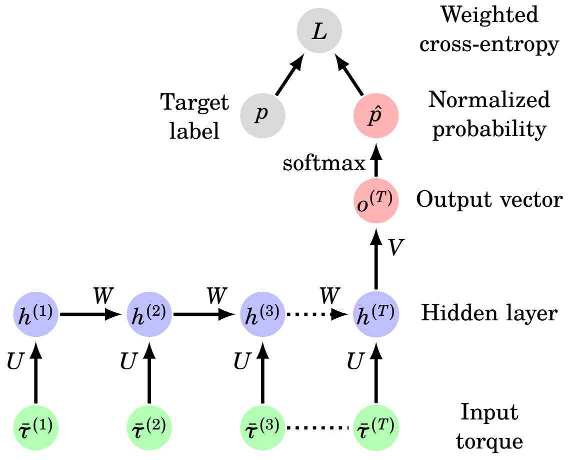


Figure 7.3 The RNN visualized as an unfolded computational graph, where each node is associated with a certain time step. The biases b and c, as well as the activation function $\tanh(\cdot)$, are omitted for a clearer view, but the computations are detailed in (7.1) to (7.3). The input torque was used to determine the hidden state, which was updated each time step. The last hidden state was used to determine the normalized probability $\hat{p}$ of whether a transient was present in the time sequence or not.

procedure was repeated 50 times, which yielded 50 positive data points. Data prior to each transient were used to form negative data points.

Given n_{pre} and n_{post}, a positive data point was formed by extracting a torque sequence, from n_{pre} samples previous to the peak value of the transient (inclusive), to n_{post} samples after (inclusive). Negative data points, with the same sequence length T as the positive ones, were extracted from torque measurements that ranged from a couple of seconds before the transient, until the positive data point (exclusive). The negative data points were chosen so that overlap was avoided. For each data point, the target was labeled as a two-dimensional one-hot vector p, where $p = [1 \;\; 0]^{\mathrm{T}}$ represented a positive data point indicating that a transient was present, and $p = [0 \;\; 1]^{\mathrm{T}}$ represented a negative data point indicating that no transient was present.

Note that with the approach above, it was possible to extract several negative data points, but only one positive data point, for every assembly trial experienced by the robot.

Half of the positive and negative data points were used in the training set, and the other half was used in the test set.

Model Training

Given the training set, the model parameters U, V, W, b, and c were determined by minimizing a loss function L. The training set contained much more negative data points than positive ones. If not taken into account, this type of class imbalance has been reported to obstruct the training procedure of several different classifiers. The phenomenon has been described in more detail in [Japkowicz, 2000; Japkowicz and Stephen, 2002], and should be taken into account when designing the loss function. Consider first the following loss function $\tilde{L}$, which is the ordinary cross-entropy between training data and model predictions, averaged over the training examples.

$$\tilde{L} = -\frac{1}{D}\sum_{d=1}^{D}\sum_{a=1}^{A} p_a^d \log\left(\hat{p}_a^d\right) \tag{7.4}$$

Here, a and d are indices for summing over the vector elements and training data points, respectively. This cross-entropy is commonly used as a loss function in machine learning [Rubinstein and Kroese, 2013; Goodfellow et al., 2016]. Because of the class imbalance in the present training set, it would be possible to yield a relatively low loss $\tilde{L}$ by simply classifying all or most of the data points as negative, regardless of the input, even though that strategy would not be desirable.

In order to take the class imbalance into account, weighted cross-entropy [Japkowicz and Stephen, 2002; Panchapagesan et al., 2016] was used as loss function. Denote by r the ratio between negative and positive data points in the training set, and introduce the weight vector $w_r = [r \quad 1]^{\mathrm{T}}$. The loss function was defined as

$$L = -\frac{1}{D}\sum_{d=1}^{D}\sum_{a=1}^{A} p_a^d \log\left(\hat{p}_a^d\right) \cdot w_r^{\mathrm{T}} y^d \tag{7.5}$$

The values of n_{pre} and n_{post} were determined using both the training set and the test set as follows. All positive data points available were used, and $r = 20$ times as many negative data points. Starting with $n_{\text{pre}} = n_{\text{post}} = 1$, the model was trained using the training set, and its performance was measured using the test set. Subsequently, both n_{pre} and n_{post} were increased by 1, and the training and evaluation procedure was repeated. This continued until perfect classification was achieved, or until the values of n_{pre} and n_{post} were large; 30 was chosen as an upper limit, though it was never reached in the experiments presented here. Thereafter, n_{pre} was kept constant, and it was investigated how much n_{post} could be lowered with retained performance. This was done by decreasing n_{post} one step at a time, while repeating the training and evaluation procedure for each value. Once the performance was decreased, the value just above that was chosen for

n_{post}. This way, the lowest possible value of n_{post} was found, that resulted in retained performance.

Once n_{pre} and n_{post} were determined, new model parameters were obtained by training on a larger data set, with $r = 100$. The reason for using a lower value for the other iterations, was that it took significantly longer computation time to use such a large data set.

Because of the class imbalance in the test set, ordinary classification accuracy, as defined by the number of correctly classified test data points divided by the total number of test data points, would not be a good model-performance measurement. Instead, the F-measurement [Hripcsak and Rothschild, 2005] was used, defined as

$$F_1 = 2\frac{PR}{P+R} \tag{7.6}$$

where P is the precision, *i.e.*, the number of correctly classified positive data points divided by the number of all data points classified as positive by the model, and R is the recall, *i.e.*, the number of correctly classified positive data points divided by the number of all data points that were truly positive. The value of F_1 ranges from 0 to 1, where 1 indicates perfect classification.

Software Implementation

The RNN in Sec. 7.3 was defined as a computational graph in the Julia programming language [Bezanson et al., 2014], using TensorFlow [Abadi et al., 2016; TensorFlow, 2019] and the wrapper TensorFlow.jl [Malmaud, 2017]. The Adam algorithm [Kingma and Ba, 2014] was used for minimization of the loss function L.

After training, the RNN model was saved on a server, and loaded into a Julia program on a PC, which communicated with the internal controller of the robot as described in Chapter 2. The sample frequency was 250 Hz, and hence the sample period was 4 ms. The robot joint torques were logged and saved, and for each time sample t, a joint-torque sequence was formed by the samples in $[t - n_{\text{pre}} - n_{\text{post}}; t]$, and sent as input to the RNN. Measurements after time t were not available at t, which is why the input only contained samples up until t. Each sequence was classified in real time. The computation time for one classification was short; well below the sample period. To move the robot, desired velocity references for the gripper in Cartesian space were first specified in the Julia program. Then, the corresponding joint velocities were computed using the robot Jacobian, and these were sent as references to the internal controller of the robot.

7.4 Experiments

Two ABB YuMi robots, shown in Figs. 7.1 and 7.4 and described in Chapter 2, were used for experimental evaluation. The implementation in Sec. 7.3 was used for robot control and transient detection. To facilitate understanding of the experimental setups and results, a video is available on [Karlsson, 2017b]. The same RNN, obtained as described in Sec. 7.3, was used for detection in each setup. The following three setups were considered.

Setup 7.1 The same robot as used for RNN training was used. Since the test set had been used to determine the hyperparameters of the RNN, *i.e.*, n_{pre} and n_{post}, it was necessary to gather new measurements to evaluate the general performance. The experimental setup was similar to that in Sec. 7.3, except that the measured torque sequences were saved and classified by the RNN, instead of just saved to the training and test sets. The robot was programmed to first move its gripper down, thus pushing the switch against the box. Once a transient was recognized, it was programmed to stop its downward motion, and instead move the box to the side. The assembly was repeated 50 times, to evaluate the robustness of the proposed approach. The experimental setup is visualized in Fig. 7.1.

Setup 7.2 The same robot as used for RNN training was used for manipulation tasks that involved objects not used during model training, in order to test its generalizability. Again, the robot was programmed to move its gripper down until a transient was detected. Once a transient was detected, it was programmed to move its gripper upwards. Three tasks were considered: snapping a battery cover into place on a pocket calculator, closing a spectacle case, and pushing a power switch on an extension cord. The setup for each task is shown in Fig. 7.2. For each of these tasks, 50 attempts were made for validation.

Setup 7.3 A different robot individual but of the same model as used for RNN training was used for manipulation of a power switch. The setup is shown in Fig. 7.4. These experiments were done at ABB Corporate Research in Västerås, Sweden. The RNN was downloaded on a local PC, and used for the detection. A total of 5 trials were done.

7.5 Results

The performance of the RNN on the test set, for different values of the hyperparameters, is shown in Table 7.2. The abbreviations are as follows: number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The hyperparameters were increased until $n_{\text{pre}} =$

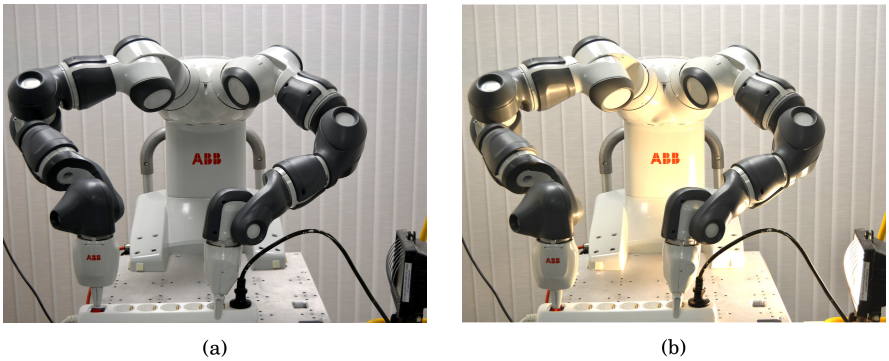


Figure 7.4 Experimental setup at ABB Corporate Research in Västerås, Sweden (Setup 7.3). The YuMi robot was different from the one used to acquire training and test data. The task began in (a) and finished in (b).

$n_{\text{post}} = 5$, for which perfect classification was obtained. Then, n_{post} was decreased until the performance decreased at $n_{\text{post}} = 2$. With this value of n_{post}, larger values of n_{pre} were tested (see second last row in Table 7.2), which did not yield perfect classification for any value, *i.e.*, $F_1 < 1$. Thus, $(n_{\text{pre}}, n_{\text{post}}) = (5, 3)$ was chosen for the final model.

After training, the RNN detected all 50 snap-fits in Setup 7.1 correctly, without any false positives prior to the snap. The torque data and RNN output from one of the trials are shown in Figs. 7.5 and 7.6. The other trials gave qualitatively similar results. The RNN also detected each transient

Table 7.2 RNN performance on the test set, for different values of the hyperparameters. The row with the lowest value of n_{post} that yielded perfect classification is marked in **blue**. With these values, the model was trained and tested again, but now with more negative data points (see last row, marked in **red**).

n_{pre}	n_{post}	TP	TN	FP	FN	P	R	F_1
1	1	20	500	0	5	1	0.80	0.89
2	2	22	500	0	3	1	0.88	0.94
3	3	23	498	2	2	0.92	0.92	0.92
4	4	23	500	0	2	1	0.92	0.96
5	5	25	500	0	0	1	1	1
5	4	25	500	0	0	1	1	1
5	**3**	**25**	**500**	**0**	**0**	**1**	**1**	**1**
5	2	22	500	0	3	1	0.88	0.94
6, 7, ... 30	2	-	-	-	-	-	-	< 1
5	**3**	**25**	**2500**	**0**	**0**	**1**	**1**	**1**

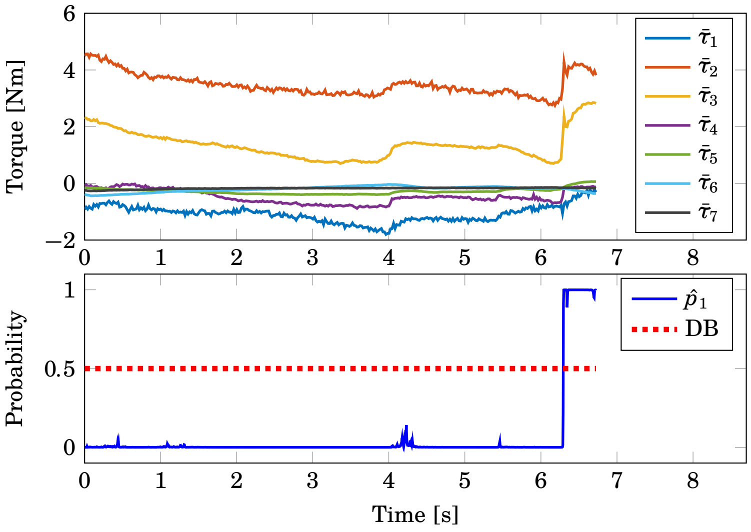


Figure 7.5 Data from Setup 7.1. The robot joint torques (upper plot) were used as input for the RNN. The torque of joint i is represented by $\bar{\tau}_i$. The first element of the RNN output ($\hat{p}_1$ in the lower plot) was close to 0 before the assembly was completed, and increased to close to 1 once the task was finished. Completion was indicated when $\hat{p}_1$ was above the decision boundary (DB, at 0.5) for the first time. Thus, for detection purposes, the RNN output generated after this event was not relevant.

successfully, without any false positives, for each trial in Setup 7.2. Data from one trial of each task are shown in Fig. 7.7, and the other trials gave qualitatively similar results. Similarly, successful detection was achieved for the 5 trials on the robot that had not been used for training (Setup 7.3). Data from one of these trials are shown in Fig. 7.8.

7.6 Discussion

There are several alternatives to RNNs for classification. Using an RNN is motivated as follows. Two less complicated models, matched filter and logistic regression, were tried initially, without achieving satisfactory performance on test data. A linear model, an SVM, and learned force thresholds for transient detection have been evaluated in [Stolt et al., 2015b]. RNNs are specialized in processing sequential data, and the torque measurements used in this work were sequential. Two properties of the RNN contribute to lower its complexity, as compared to a standard artificial neural network. These consist of parameter sharing, which means that the same parame-

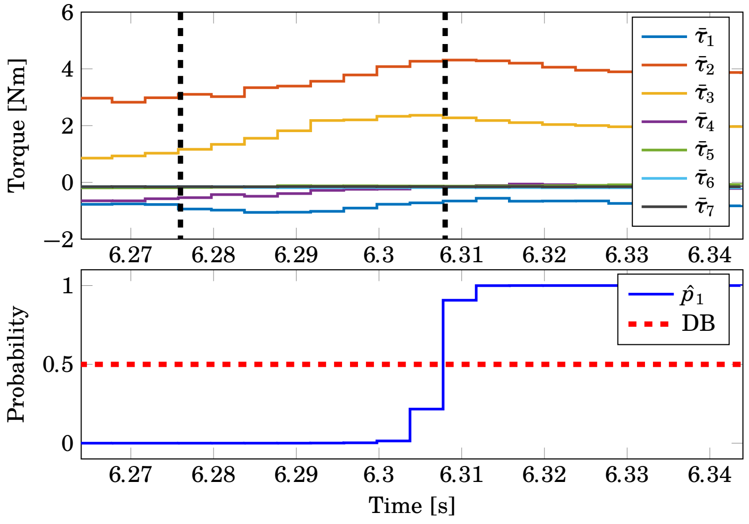


Figure 7.6 Same data as in Fig. 7.5, but zoomed in on the time when the task finished. The transient was detected at time $t = 6.308$ s. The first torque sequence to be classified as positive was that within the vertical dashed lines in the upper plot. Figure 7.5 might give the impression that $\hat{p}_1$ increased from 0 to 1 instantly upon detection. In this figure, however, it can be seen that the confidence was actually built up during a couple of sample periods, which was also the case for the remaining trials.

ters are used in several connections, and sparsity, which means that only some of the neurons are connected to each other. This can be seen in Fig. 7.3. Thanks to the lower complexity, it is possible to estimate a model with significantly fewer training examples than would be needed otherwise. Compared to models that are not specialized in sequential data, *e.g.*, logit models, SVMs, and ordinary neural networks, the RNN is less sensitive to variations of the exact time step in which some information in the input sequence appears. An RNN can also be generalized to classify data points of sequence lengths not present in the training set, though this was not used in this present work. General properties of RNNs are well described in [Goodfellow et al., 2016].

The concept of RNNs is well known from previous research [Graves, 2012; Goodfellow et al., 2016], and hence it should not be seen as a contribution from this chapter. Instead we have used it to evaluate whether transients in robot motor torques could be used for validation of manipulation tasks, which in turn is the main purpose of this chapter. The RNN architecture described in Sec. 7.3 serves as an example of how joint torques could be used for validation in practice.

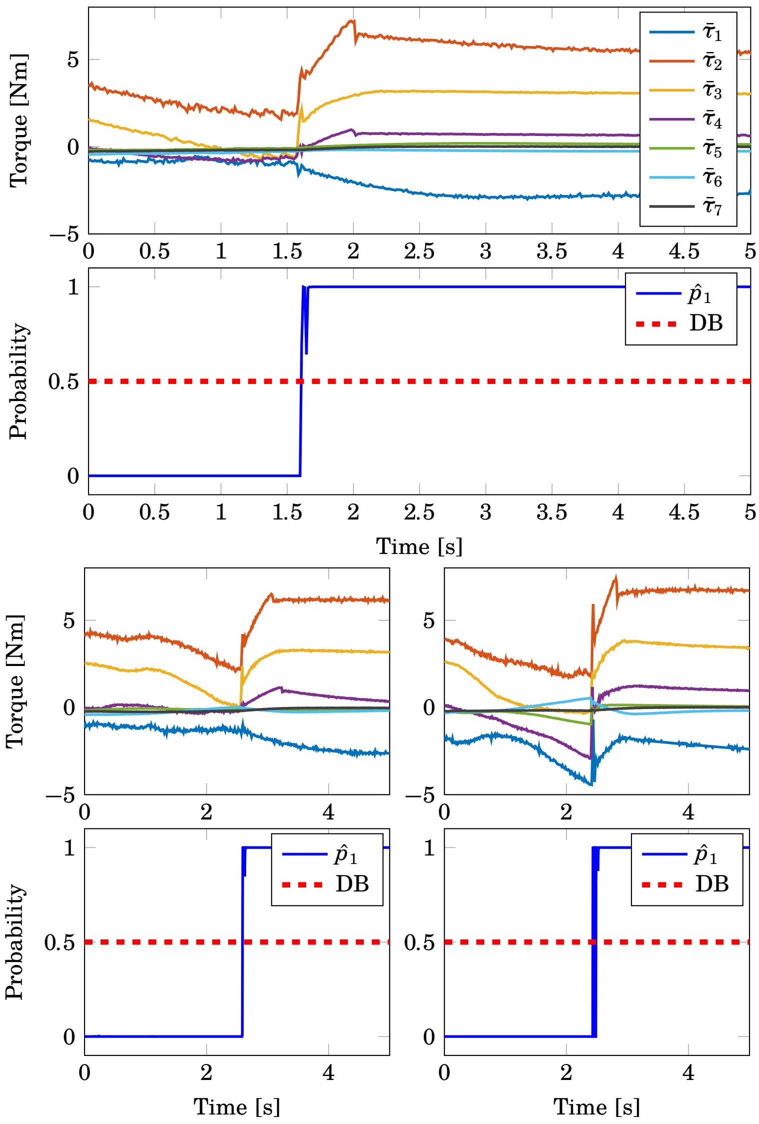


Figure 7.7 Experimental data from the manipulation tasks in Setup 7.2; snapping the battery cover into place on the pocket calculator (upper), closing the spectacle case (lower left), and pushing the power switch on the extension cord (lower right). The organization of each plot is the same as in Figs. 7.5 and 7.6.

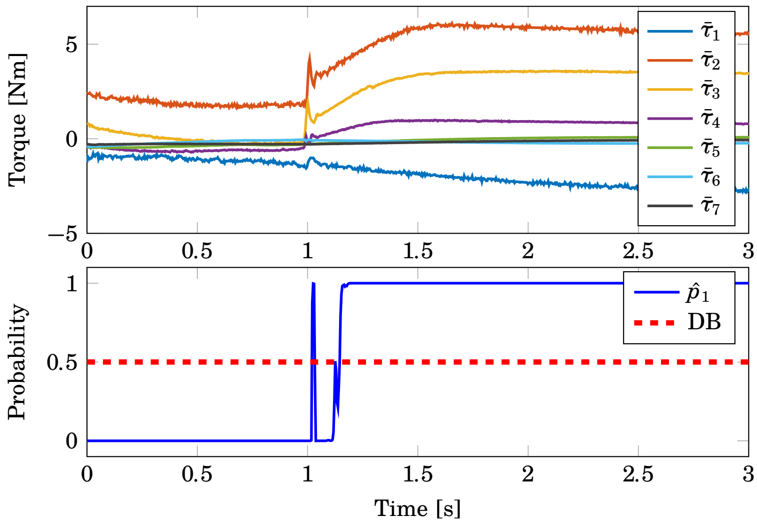


Figure 7.8 Experimental data from Setup 7.3, where a power switch on an extension cord was pushed by a YuMi robot not used for RNN training. Detection occurred at time $t \approx 1\,\mathrm{s}$.

As compared to [Stolt et al., 2015b], the approach proposed here had four new benefits. A force sensor was no longer required, the detection delay was reduced, the computation time for model training was shortened, and the confidence level for detection could be set through one parameter. In [Stolt et al., 2015b], $n_{\text{post}} > 10$ (corresponding to $> 40\,\text{ms}$) was required for perfect classification, whereas our proposed method required $n_{\text{post}} = 3$ (12 ms). The training time of the RNN was in the order of minutes on an ordinary PC, which was an improvement compared to days in [Stolt et al., 2015b]. The decision boundary for $\hat{p}_1$ in Sec. 7.3 corresponds to a desired confidence level for detection. It was set to 0.5 in Sec. 7.3, meaning that a task was classified as completed when the estimated probability of completion was above 50 %. However, the meaning of $\hat{p}_1$ is intuitive, and the decision boundary can easily be adjusted to a desired level of confidence. For example, it might be worse to fail to detect a force transient and risk to damage the hardware, than to finish a subtask too early and possibly detect this mistake later on. Such a cost asymmetry would motivate a lower decision boundary. In some scenarios, however, backup detectors may confirm completion reliably but late, and it might then be more important to avoid false positives. This was the case in the experimental scenario in [Stolt et al., 2015b]. The ability to adjust the confidence level according to specific circumstances is valuable. In some approaches, the confidence level is acquired by weighting false positive and false negative points differently in the cost function used in the model training, see, *e.g.*, [Stolt et al., 2015b]. This requires a new training and validation procedure to adjust the confidence level, which is avoided with the method proposed here.

Further, this research extended [Stolt et al., 2015b; Karlsson, 2017e] by verifying generalization of the classification model to manipulation of new objects, that had not been used to generate training and test data. It is expected that this generalization is limited to similar tasks. Therefore, the idea is that the training procedure described in Sec. 7.3 should be gone through whenever a detection model for a completely different task is required.

Given a certain contact force/torque acting on the end-effector of the robot, the corresponding motor torques depend on the configuration, as well as gravity and friction between arm side and motor side of each joint. It is expected that different robot individuals of the same model would experience slightly different motor torques for a given task, because of small uncertainties in the robot manufacturing and different wear and tear. In this chapter, it has been shown that generalization of the RNN to a new robot individual is possible. This fact was also used during the final review of the SARAFun project [SARAFun, 2019], where Setup 7.3 was used to demonstrate the proposed detection algorithm.

However, changing robot model or configuration would change the joint torques significantly, and generalization can not be expected with this current implementation. Figure 7.9 shows the success rate for the setup in Fig. 7.4, but where the task was carried out at different distances from the position used during training. As expected, the success rate deteriorated with increased distance. The failures consisted both of false detections and undetected transients. However, already this approach allowed for some uncertainty in the work-space configuration, in the order of 0.1 m. Even though generalization to the specific configurations tested in Fig. 7.9 could be achieved by first including those configurations in the training procedure, it would not be a feasible strategy for generalization to the entire work space. The reason is that this would require more training examples than one could provide in reasonable time.

To enable generalization also to new configurations and robot models, it would be a good idea to first model and compensate for gravity and friction-induced motor torques, then estimate the external contact forces/torques, and subsequently use these as input for the detection model. Furthermore, if external force/torque sensors are available in a given setup, it would be straightforward to include those measurements in this approach. Generalization to new robots and configurations would then easily be achieved. Motor torques could still be relevant for validation of the external measurements.

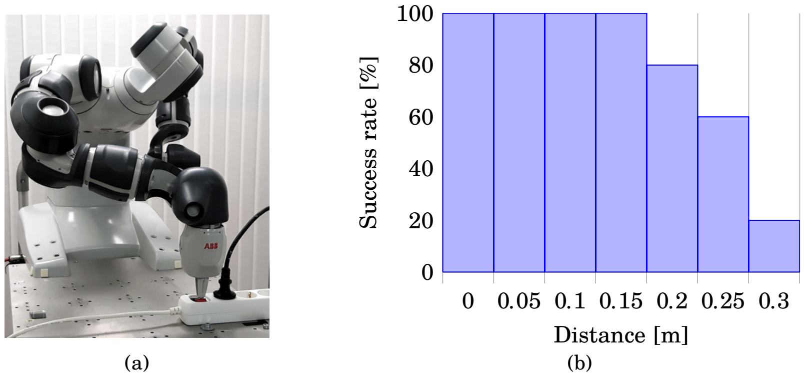


Figure 7.9 Evaluation of success rate depending on distance from work-object position used during training. The setup is shown in (a), where the task was performed 0.3 m away from the position used during training. Furthermore, the robot and the work object were different from those used during training. The success rates for different distances are shown in (b). Five trials were done for each distance.

In this architecture, task completion was concluded the first time $\hat{p}_1$ was greater than the decision boundary. Therefore, the values of $\hat{p}_1$ after that event were not relevant for detection purposes, and torque measurements after any transient were not used as training or test data. Further, the torques after task completion depend not only on the interaction forces, but also on the commanded behavior of the robot.

The concept of weighted loss to compensate for class imbalance in machine learning has been evaluated in [Japkowicz and Stephen, 2002], and successfully applied to a deep neural network in [Panchapagesan et al., 2016]. The loss function (7.5) extends (7.4) by the factor $w_r^{\mathrm{T}} y^d$, which evaluates to r for positive data points, and to 1 for negative ones.

In Fig. 7.5, it should be noted that $\hat{p}_1$ raised significantly above 0 at $t \approx 4.2$ s, even though the task did not yet finish at that time. Even though the values were still well below the decision boundary, this leaves room for improvement in terms of reliability of the detection.

In a complete setup, the validation procedure described here could be combined with more sophisticated motion controllers, such as DMPs. Whereas the evaluation of the model was done in real time in this work, the RNN training was performed offline. A complete product should have a user interface allowing for an operator to gather training data and test data, label them, and run the training procedure. This would shorten the time required for acquisition of training and test data, which was approximately one working day with the current arrangement. It would also be valuable if labels could be suggested automatically. This could for instance be achieved using a simple classifier trained on some of the first labels, which would have to be determined manually. It is also important to shorten the reaction time of the robot, *i.e.*, to further reduce the value of n_{post}. This could be done by including more sensors. For instance, the snap-fit assembly generates a sound, easily recognized by a human. Adding a microphone to the current arrangement would therefore add information to the detection approach. In turn, this could be used to decrease the required amount of training data, improve robustness of the detection, or detect the transient earlier.

7.7 Conclusion

In this research, we have addressed the question of whether robot joint-torque measurements could be used for detection of force/torque transients in robotic manipulation. We have shown that such detection is possible, which is the main contribution of this chapter. In contrast, the concept of RNNs is well known from previous research, and was used in this chapter for the purpose of evaluation, as well as to exemplify how joint torques could be used for detection in a practical setup. First, training and test data were gathered and labeled. Then, these were used to determine the parameters of the detection model. Finally, a real-time application for transient detection was implemented and tested, both on the assembly task used to generate training and test data, and on new tasks and using a new robot, that had not been used to determine any model parameters. The method presented seems promising, since the resulting model had high performance, both on the test data and during the experiments. A video that shows the functionality is available on [Karlsson, 2017b].

8

A Dual-Arm Haptic Interface for Task Demonstration

8.1 Introduction

Whereas purely position-based programs can be mediated to robots through ordinary lead-through programming, tasks that involve contact forces between a robot and its surroundings are not as straightforward to demonstrate. Ordinary lead-through programming, using only one robot arm, yields forces between the operator and the robot which can not be distinguished from the task-specific forces in general. In this chapter, an immersive haptic interface for task demonstration is presented and evaluated. The interface allows for an operator to act and sense remotely, thereby demonstrating desired movements and interaction forces. The interface consists of two robot arms, one of which can be moved directly by an operator through physical contact, and one that moves accordingly, while contact forces are reflected to either arm as haptic feedback. A control law enforces a constant pose offset in Cartesian space between the end-effectors of the robot arms. We refer to the robot arm in direct contact with the operator as the master side, and to the other robot arm as the slave side. However, the presented algorithm is symmetric with respect to the robot arms, *i.e.*, either arm could be used as master or slave without any specification. The concept is visualized in Fig. 8.1 and in a video available online [Ghazaei Ardakani et al., 2019].

Haptic feedback for robotic teleoperation [Hokayem and Spong, 2006] and for virtual reality [Constantinescu, 2008] has become an important research topic. Four-channel haptic systems represent the most general form [Aliaga et al., 2004], where position and force data are transferred between the master and slave sides in both directions. In addition to a

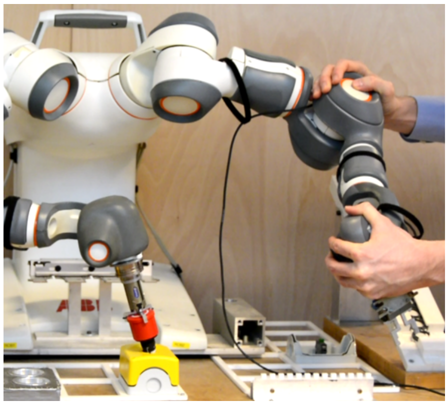


Figure 8.1 Example use case of the proposed interface. An operator demonstrates a peg-in-hole task by moving the left robot arm through physical contact. The right robot arm follows the demonstrated movement in real time.

haptic system, visual feedback from the slave side to the master side is typically necessary. In the arrangements considered in this chapter, the slave side can be seen directly by the operator, see Fig. 8.1. If this is not the case, cameras on the slave side are commonly used. A teleoperated interface is transparent if movement and contact forces are reflected to either side through the interface, so that the operator has the feeling of interacting with the remote environment.

There are several factors that limit the transparency. In [Lawrence, 1993], it has been shown that transparency and stability are conflicting features. Delays are detrimental both to stability and transparency [Anderson and Spong, 1989; Lee and Spong, 2006; Hatanaka et al., 2015]. Light-weight collaborative robots may be used as haptic devices without external sensors. Joint friction should then be compensated for, and uncertainties in the friction model would limit the transparency. In this chapter, we describe how measurements of external forces can be included in the control law, but in the experiments we consider a worst-case scenario where external sensors are not available. Transparency is further limited by the structures of the master and slave devices. The common work space is limited to the configurations reachable by both devices simultaneously. At the workspace boundary, at least one device has reduced manipulability because of a singular configuration or limits of reach or joint angles. It is desirable to provide feedback regarding these internal limitations, though this contradicts transparency with respect to the environment. The operator typically expects force feedback both from limitations of the robotic system, and from interaction with the environment. The proposed algorithm supports deployment of two not-necessarily-similar robots, and the ability to reflect the internal limitations of the system then becomes of major importance.

The main contribution of this chapter is the implementation and evaluation of a haptic interface between two not-necessarily-similar robot arms, which can be interacted with at any point of their structures. Whereas in previous methods, the master and slave devices have commonly been mounted and configured to avoid difficulties in case of dissimilarities between their individual work spaces, this proposed method supports use of any six-degree-of-freedom or redundant robot arms independently, mounted on any surfaces and with any initial configurations. It further results in theoretically perfect pose tracking during free-space motion, and perfect force tracking in static contact with the environment except at the boundary of the common work space where internal limitations are sensed by the operator. Any of the arms can pass through singular configurations while the haptic feedback is adjusted accordingly. The control law is based on virtual constraints, and suits well for kinesthetic teaching. The robot arms are coupled in Cartesian space to retain a constant pose offset between their end-effectors. This chapter is based on our previous publication [Ghazaei Ardakani et al., 2018], in which we extended the research in [Ghazaei Ardakani, 2016] by implementing the proposed algorithm and evaluating it experimentally. Simulation results are available in [Ghazaei Ardakani, 2016].

8.2 Problem Formulation

It is difficult to program robots for tasks that require force interaction, because both knowledge of the specific task and the robot dynamics are necessary. Whereas positions can be visualized and specified graphically in user interfaces, interaction forces are not directly visible [Johansson et al., 2015]. A human typically does not know required interaction forces in quantitative terms as physical entities, but may still be able to perform and demonstrate the task. Programming by demonstration is a natural way for humans to mediate behavior to robots [Lee et al., 2015]. If interaction forces could be extracted from a demonstration, intuitive specification of required forces would be achieved. Lead-through programming [Pan et al., 2010; Stolt et al., 2015a; Capurso et al., 2017] has the benefit that it allows for the operator to perceive the mechanical properties and limitations of the robot during the demonstration. Unfortunately, however, ordinary lead-through programming implies contact forces between the operator and the robot, which are in general impossible to distinguish from the demonstrated interaction forces between the robot and the work objects.

In this chapter, we therefore implement a dual-arm haptic interface for task demonstration, where one robot arm can be moved directly by the operator, and the other robot arm moves accordingly. Because interaction with the operator and with the work objects have been separated to different robot arms, demonstrated forces between robot and work objects are easily distinguished. In the context of task demonstration through a haptic interface, transparency is an important aspect. Further, it is important to consider stability, especially during physical contact with a stiff environment. Therefore, this chapter also aims to evaluate the movement and force tracking between the two robot arms, as well as the stability of the proposed system.

8.3 Previous Research

A survey of haptic feedback in teleoperation is given in [Hokayem and Spong, 2006]. There are currently several approaches to coupling a slave and a master device. In [Rebelo and Schiele, 2012], the placement of the slave device was optimized to maximize manipulability, and position offset and scaling were used to avoid work-space limitations. A point-to-point kinematic mapping to maximize the common work space was presented in [Chen et al., 2007]. The trajectory of the master robot was mapped into positions that were reachable for the slave robot.

A passive control law for teleoperation in joint space, capable of inertia scaling and obstacle avoidance, was proposed in [Lee and Li, 2005]. Virtual-constraint forces have been introduced using potential fields, which can be used to help the operator to remain in a region of interest, and avoid obstacles and singularities [Turro et al., 2001]. Furthermore, force feedback from virtual damped springs have been introduced in virtual-reality applications [Constantinescu et al., 2006], to simulate contact with the virtual environment.

In the context of kinesthetic teaching, it is desirable to compensate for joint friction. Advancements in sensorless force estimation have been reported in [Linderoth et al., 2013; Wahrburg et al., 2014; Stolt et al., 2015c], and lead-through programming using friction compensation has been developed in [Stolt et al., 2015a; Ghazaei Ardakani, 2016; Capurso et al., 2017].

The control law described in this chapter is synthesized by constraining the poses of the robot end-effectors virtually. Virtual constraints have been used for control synthesis in previous research, see, *e.g.*, [Shiriaev et al., 2007; Westervelt et al., 2007; Freidovich et al., 2008]. Further, several researchers have taken the view of a teleoperation system as a passive mechanical tool [Itoh et al., 2000; Lee and Li, 2003], which is also done in this chapter.

Table 8.1 Notation used in this chapter. The subscripts 1 and 2 are used to distinguish each robot arm. The subscript i indicates one arbitrary arm.

Notation		Description
n_i	$\in \mathbb{Z}^+$	Number of joints on arm i
n	$\in \mathbb{Z}^+$	$n_1 + n_2$
q	$\in \mathbb{R}^n$	Joint positions
p_i	$\in \mathbb{R}^3$	Cartesian position
R_i	$\in \mathrm{SO}(3)$	Cartesian orientation
J_i	$\in \mathbb{R}^{6\times n_i}$	Robot Jacobian
G	$\in \mathbb{R}^{6\times n}$	$[-J_1 \;\; J_2]$
M	$\in \mathbb{R}^{n\times n}$	Mass matrix
C	$\in \mathbb{R}^{n\times n}$	Centripetal and Coriolis matrix
Q^{e}	$\in \mathbb{R}^n$	External torques
Q^{m}	$\in \mathbb{R}^n$	Motor torques scaled to arm side
Q^{vc}	$\in \mathbb{R}^n$	Torques due to virtual constraint
λ	$\in \mathbb{R}^6$	Forces due to virtual constraint
e	$\in \mathbb{R}^6$	Pose deviation from offset constraint
$\dot{y}$	$\in \mathbb{R}^6$	Difference between end-effector velocities
Γ	$\in \mathbb{R}^{6\times 6}$	$GM^{-1}G^{\mathrm{T}}$
x, y, z	$\in \mathbb{R}$	Cartesian coordinates, see Fig. 8.3

8.4 Method

The control law for the haptic interface implemented in this chapter was first described in [Ghazaei Ardakani, 2016], and is briefly described in this section for convenience. The main idea is to virtually constrain the two end-effectors to have a constant pose offset, as if they were physically and rigidly attached to each other, and subsequently design a control law that realizes this constraint. Table 8.1 lists some of the notation used in this chapter.

Virtual Constraint

Denote by n_1 and n_2 the number of joints on the first and second arm, respectively, and let $n = n_1 + n_2$ represent the total number of joints. Throughout this chapter, we will continue to use the subscripts 1 and 2 to associate quantities with each of the two arms. Further, let $q_1 \in \mathbb{R}^{n_1}$ and $q_2 \in \mathbb{R}^{n_2}$ represent joint positions, and introduce $q = [q_1^{\mathrm{T}} \;\; q_2^{\mathrm{T}}]^{\mathrm{T}}$. The geometric constraint between the end-effectors can be expressed as

$$p_2 - p_1 = \Delta p \tag{8.1a}$$

$$R_1^{\mathrm{T}} R_2 = \Delta R \tag{8.1b}$$

where $p_i \in \mathbb{R}^3$ represents position, $R_i \in \mathrm{SO}(3)$ represents orientation, and Δp and ΔR are constant offsets in position and orientation, respectively. For the control design, Δp and ΔR will be seen as references for the pose offset.

THEOREM 8.1
The geometric constraint in (8.1) implicates the kinematic constraint given by

$$G\dot{q} = 0 \tag{8.2}$$

where $G = [-J_1 \ \ J_2] \in \mathbb{R}^{6\times n}$ consists of the robot Jacobians J_i. □

Proof. See Sec. 8.A. □

Control Algorithm

The equations of motion for a rigid-body model of each robot arm i can be written as [Spong et al., 2006]

$$M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i = Q_i^{\mathrm{e}} + Q_i^{\mathrm{m}} - \mu_i \dot{q}_i \tag{8.3}$$

where torques induced by Coulomb friction and gravity have been omitted since we compensate for these separately. Here, $M_i(q_i)$ is the mass matrix, $C_i(q_i, \dot{q}_i)$ represents the effect of centripetal and Coriolis forces, Q_i^{e} represents external torques, Q_i^{m} represents motor torques scaled to the arm side, and μ_i represents coefficients for viscous friction. Denote by Q^{vc} motor torques that enforce the virtual constraint. Our aim is to use these as control signals, so that $Q^{\mathrm{m}} = Q^{\mathrm{vc}}$. The equations of motion of the constrained system can be derived from (8.2) and (8.3) as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = Q^{\mathrm{e}} + Q^{\mathrm{vc}} - \mu \dot{q} \tag{8.4a}$$

$$G\dot{q} = 0 \tag{8.4b}$$

where

$$M(q) = \mathrm{blkdiag}\,(M_1(q_1), M_2(q_2)) \tag{8.5a}$$

$$C(q, \dot{q}) = \mathrm{blkdiag}\,(C_1(q_1, \dot{q}_1), C_2(q_2, \dot{q}_2)) \tag{8.5b}$$

Furthermore, we introduce forces at the end-effectors that enforce the virtual constraint, denoted by λ. These are related to the motor torques due to the virtual constraint as follows.

$$Q^{\mathrm{vc}} = G^{\mathrm{T}}\lambda \tag{8.6}$$

Intuitively, $-\lambda$ and λ can be interpreted as interaction forces between the end-effectors, if they were physically attached to each other. The interaction forces have equal magnitude and opposite direction, as predicted by Newton's third law of motion. The opposite directions are taken into account by the different signs of the Jacobians in G, so that

$$Q^{\mathrm{vc}} = G^{\mathrm{T}}\lambda = \begin{bmatrix} -J_1^{\mathrm{T}} \\ J_2^{\mathrm{T}} \end{bmatrix} \lambda = \begin{bmatrix} J_1^{\mathrm{T}}(-\lambda) \\ J_2^{\mathrm{T}}\lambda \end{bmatrix} \tag{8.7}$$

By introducing the state vector $\xi = [q^{\mathrm{T}} \;\; \dot{q}^{\mathrm{T}}]^{\mathrm{T}}$ and replacing λ with u, (8.4) can be written as

$$\dot{\xi} = \begin{bmatrix} \dot{q} \\ M^{-1}(q)\left(-C(q,\dot{q})\dot{q} + Q^{\mathrm{e}} - \mu\dot{q} + G^{\mathrm{T}}(q)u\right) \end{bmatrix} \tag{8.8a}$$

$$\dot{y} = G(q)\dot{q} \tag{8.8b}$$

where $\dot{y}$ denotes the difference between the end-effector velocities. Define $\Gamma = GM^{-1}G^{\mathrm{T}}$. The control signal $u^*(\xi)$ that yields zero relative acceleration, *i.e.*, $\ddot{y} = 0$, can be obtained by differentiating (8.8b) with respect to time and solving for u which yields

$$u^* = \Gamma^{-1}\left(GM^{-1}\left(C\dot{q} - Q^{\mathrm{e}} + \mu\dot{q}\right) - \dot{G}\dot{q}\right) \tag{8.9}$$

This solves the servo problem. To also solve the regulation problem, *i.e.*, to drive the physical offset to the reference in case of any deviation, we introduce the state feedback

$$\lambda = u = u^* - \Gamma^{-1}\left(K_{\mathrm{d}}\dot{y} + K_{\mathrm{p}}e\right) \tag{8.10}$$

This results in asymptotic stability of (8.8) [Ghazaei Ardakani et al., 2018]. Further, if one of the arms is in contact with the environment and standing still, and no internal limits of the robots are reached, the forces acting on the end-effectors are theoretically perfectly transferred [Ghazaei Ardakani et al., 2018]. Here, K_{p} and K_{d} are positive definite gain matrices. These matrices are block diagonal, *i.e.*, $K_{\mathrm{p}} = \mathrm{blkdiag}(K_{\mathrm{tp}}, K_{\mathrm{op}})$, where K_{tp} is a positive definite gain matrix for translation, and K_{op} is a positive definite gain matrix for orientation. The matrix K_{d} has a corresponding structure. Further, e is the pose difference between the end-effectors beyond the desired offset, *i.e.*, $e = [e_{\mathrm{P}}^{\mathrm{T}} \;\; e_{\mathrm{O}}^{\mathrm{T}}]^{\mathrm{T}}$, where $e_{\mathrm{P}} = p_2 - (p_1 + \Delta p)$ and e_{O} is the imaginary part of the unit quaternion corresponding to $R_2(R_1\Delta R)^{\mathrm{T}}$. Hence, e represents an error which is 0 when the geometric constraint in (8.1) is fulfilled. The error dynamics can be obtained by inserting the control law (8.10) into the equations of motion (8.8) which yields

$$\ddot{y} + K_{\mathrm{d}}\dot{y} + K_{\mathrm{p}}e = 0 \tag{8.11}$$

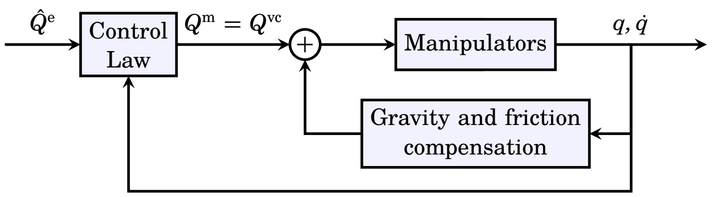

$\hat{Q}^{\mathrm{e}}$
Control Law
$Q^{\mathrm{m}} = Q^{\mathrm{vc}}$
+
Manipulators
$q, \dot{q}$
Gravity and friction compensation

Figure 8.2 Control architecture. Estimated external torques $\hat{Q}^{\mathrm{e}}$ and the states of the robots are inputs to the controller for the calculation of Q^{vc}, the torques from the virtual constraint. The manipulators are gravity and Coulomb-friction compensated.

The motor-torque references sent to the robots are given by

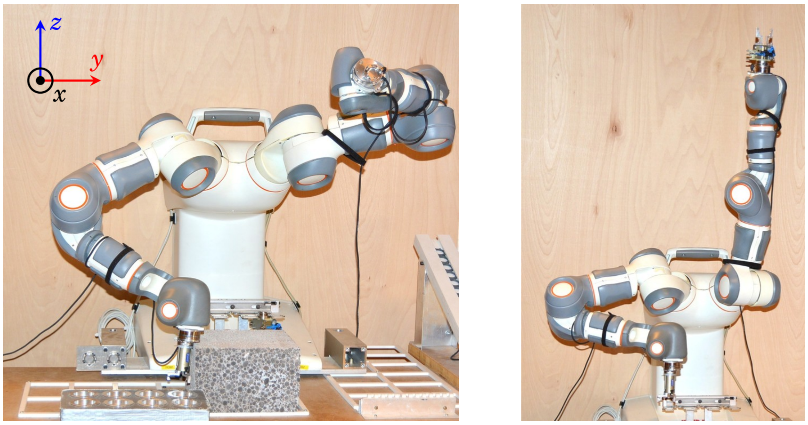
$$Q^{\mathrm{m}} = Q^{\mathrm{vc}} = G^{\mathrm{T}}\lambda \tag{8.12}$$

The control architecture is visualized in Fig. 8.2.

8.5 Experiments

For the experiments, the ABB YuMi prototype was used. The control algorithm in Sec. 8.4 was implemented on a PC that communicated with the internal controller of the robot. According to Fig. 8.2, the sum of the generalized forces from the virtual constraint Q^{vc}, and the gravity and friction compensation signals, was sent as a torque reference to the internal robot controller. For the friction compensation at zero joint velocities, a dithering technique was used [Capurso et al., 2017]. The worst-case scenario was experimentally evaluated, where no external sensors were assumed and the estimated external forces and torques were set to zero. Each end-effector of the robot was equipped with a force/torque sensor, and these were used for evaluation only. The experimental arrangement is shown in Fig. 8.3. Additionally, a video that shows the setup and the experiments is available online [Ghazaei Ardakani et al., 2019]. Two different setups were considered.

Setup 8.1 The left robot arm was used as master. First, free-space motion was conducted, in order to evaluate the motion tracking between the two arms. The right arm was then brought to collision with a concrete block in front of the robot, in order to evaluate force tracking and stability in contact with a stiff environment. When in contact, the operator pushed the end-effector of the master arm in the positive y-direction, defined in Fig. 8.3, so that the slave arm was pushed against the block in order to excite the interaction forces.



(a) (b)

Figure 8.3 Robot configurations for experimental evaluation. Setup 8.1 is shown in (a), where the left arm of the robot was used as master, and the movement was such that the end-effector of the right arm established contact with the concrete block in front of the robot. Setup 8.2 is shown in (b), where the right robot arm was used as master, and the left arm was first brought upwards to its reach limit.

Setup 8.2 The right robot arm was used as master. By moving the master arm, the left arm was first brought to its upper reach limit. This configuration was also a singularity, see Fig. 8.3. Thereafter, it was moved slightly to the side thus reaching the angle limit of Joint 1, the joint closest to the body. The purpose of this experiment was to evaluate stability properties and interaction forces when the slave arm reached internal physical limitations.

8.6 Results

Data from Setup 8.1 are shown in Fig. 8.4. It can be seen that motion tracking was achieved. The bottom plot compares the forces in the y-direction, as measured by the wrist-mounted force sensors. The force measured on the left arm has been negated for easier comparison, and corresponds to the interaction force experienced by the operator. After 50 s of free-space motion, a contact was established between the right end-effector and the concrete block. It can be seen that force tracking was achieved, meaning that the operator experienced the same contact forces as the slave arm. Moreover, the contact force from the concrete block prevented both arms from moving further in the y-direction, despite the force exerted by the operator. This behavior is the desired result of the virtual constraint.

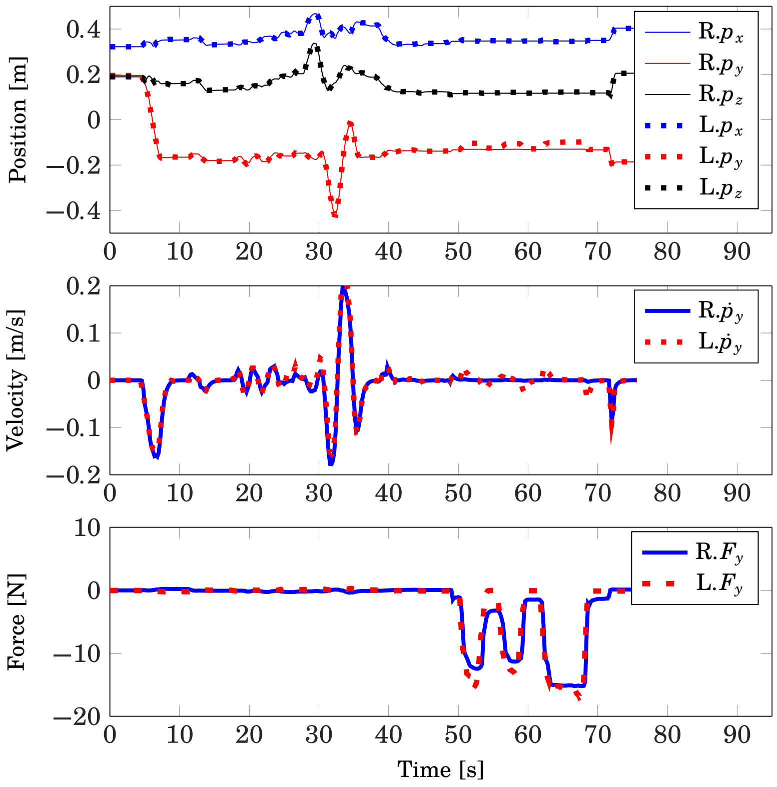


Figure 8.4 Results from Setup 8.1. For an uncluttered view, only some of the dimensions are shown. The first plot from above shows the position of the arms, where the position of the left arm has been compensated for with the desired offset between the arms. The second plot shows velocities in the y-direction, and the third plot shows forces in the y-direction. R.p_x represents the right arm's position in the x-direction, and so on.

Data from Setup 8.2 are visualized in Fig. 8.5. The left arm was first brought to its upper reach limit. Despite a large force exerted upwards on the master arm by the operator, it can be seen that the master arm was prevented from moving in the z-direction by the virtual constraint. Joint 1 of the left arm was thereafter brought to its angle limit. Then, further motion of the arms in the limited direction was prevented despite pushing the master arm, since the joint limit of the slave arm was being propagated to the master arm thanks to the virtual constraint.

As expected from an impedance-type controller, stability was retained both during free-space motion, during contact with a stiff environment, and

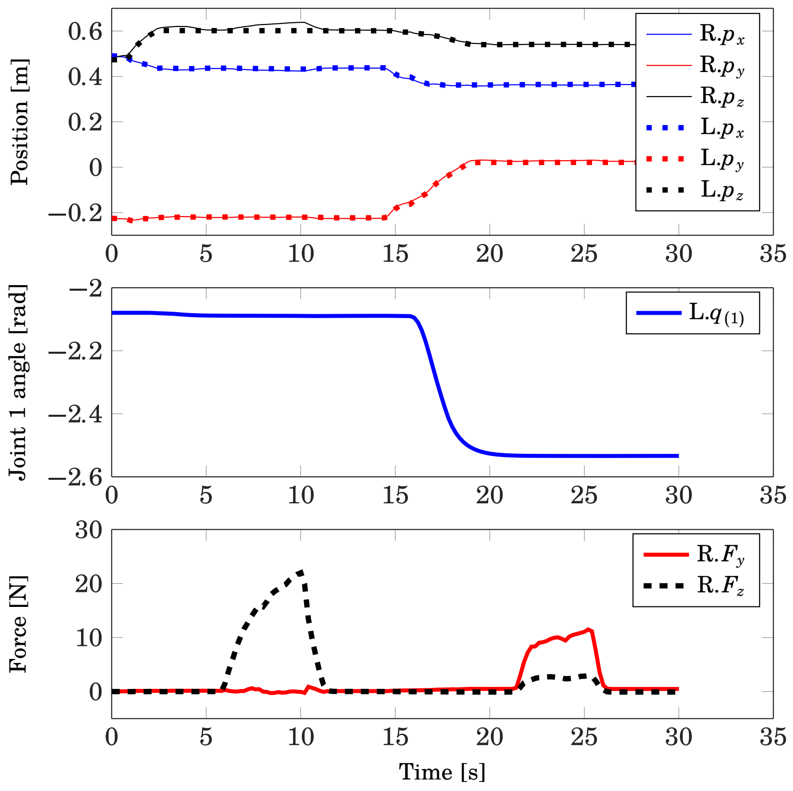


Figure 8.5 Results from Setup 8.2. The master arm was subjected to large forces twice; first when the left arm reached a work-space limit in the z-direction, and thereafter when Joint 1 of the left arm was in its lower angle limit.

while reaching stiff internal limitations. In addition, it can be concluded that the external physical limitation caused by the concrete block in Setup 8.1 had a similar effect as the internal limitations in Setup 8.2; the motion of the slave arm was limited directly, and the master arm was limited through the virtual constraint. Moreover, the virtual constraint transferred the contact forces between the arms, giving the operator the feeling of interacting with the environment using a passive tool.

8.7 Discussion

The proposed control law solves the problem of capturing interaction forces from demonstrations based on lead-through programming. The dual-arm lead-through programming is more user friendly and has less side effects than single-arm arrangements, thanks to the separation between contact forces from the operator and the work objects.

A possible drawback with the proposed approach is that the physical dynamical properties of the robots may be undesired. This can be mitigated by using local feedback to adjust the apparent dynamical properties, before applying the proposed controller.

In the current sensorless implementation, the force tracking was mainly limited by uncertainties in the friction model. A simple model was adopted, consisting of Coulomb and viscous friction. In reality, many aspects affect joint friction, such as load, configuration [Wahrburg et al., 2018], and temperature [Bagge Carlson et al., 2015]. Especially zero-velocity friction is difficult to estimate [Stolt et al., 2015a; Capurso et al., 2017]. In the experiments, external sensors were not used for the haptic interface, to investigate a worst-case scenario. The algorithm supports integration of external sensors, which would improve the overall performance significantly. The sensorless approach is limited to light-weight collaborative robots, as large industrial robots exhibit more joint friction and usually require external sensing for any lead-through programming.

The prototype YuMi robot was used partly because it had force sensors for validation, and partly because its research interface was more mature than that of the product version of YuMi at the time, with predictable delays and low jitter, which was important for the implementation.

This research has implications for the industry, as the robot company Cognibotics AB, Lund, Sweden, aims to commercialize the interface. As a future step, the implementation will be migrated to the product version of YuMi, possibly using another version of the research interface. It will then be a benefit that the product version of YuMi exhibits lower and more predictable joint friction than the prototype.

Using a dual-arm robot for this interface has the benefit that two arms are then naturally available. In such an arrangement, however, manipulation tasks that require both arms leave no arm available as master side. This can be solved by using yet another robot as master. For instance, it would be straightforward to use an entire YuMi robot as master, controlling all 14 degrees of freedom, and another one as slave. Enforcing the same configuration on both robots would then be possible and perhaps more intuitive for an operator, but would not be a requirement for the interface as it also supports independent configurations of the master and slave.

The control signal in (8.9) results in zero dynamics of the system (8.8). It was found by following the steps described in [Slotine and Li, 1991], *i.e.*, time differentiate the output until the control signal appears, and subsequently choose the control signal to cancel nonlinearities and guarantee tracking. Alternatively, the control signal for the zero dynamics could be derived using the general results in [Isidori, 1995], as done in [Ghazaei Ardakani, 2016; Ghazaei Ardakani et al., 2018].

The interface also allows for humans to interact remotely with each other. Early in the video in [Ghazaei Ardakani et al., 2019], two humans shake hands remotely through the interface. This result highlights the fact that the implementation is symmetric with respect to the robot arms, and that the distinction between master and slave side is artificial. Haptic feedback and physical remote human–human interaction is likely to become important in, *e.g.*, the video-game industry. Today, phone calls and video calls are frequently used instead of physical meetings. Reducing transportation needs could potentially save time and money, while decreasing the negative impact on the environment. While the demand for faster and more efficient transportation is high in general, in theory ideal telepresence could replace almost any transportation of humans.

In the presented implementation, the robot arms were just virtually constrained by the constant pose offset between their end-effectors. However, additional constraints are easy to append. In the last part of the video in [Ghazaei Ardakani et al., 2019], a virtual plane prevented the robot arms from moving too low, thus avoiding collision. In [Ghazaei Ardakani et al., 2018], it was shown how to add additional virtual constraints for redundant robots, to allow for impacting all the degrees of freedom from either side. Coupling between points other than the end-effectors, locking of certain dimensions, and scaling can also be included to further assist an operator. See, *e.g.*, [Shiriaev et al., 2007; Freidovich et al., 2008] for previous research on control design based on virtual constraints.

It would also be possible to use robot arms with less than six degrees of freedom, though this would prevent movement in six degrees of freedom for both end-effectors. Such a limitation would be reflected as haptic feedback to the other side of the interface, just like joint and reach limits as well as singularities were reflected in the current implementation.

In Chapter 7, joint torques from interaction forces were used as training data for a recognition model. To gather the training data, the robot had been programmed in advance to make a simple movement, so that the task would finish. The interface in this chapter offers the possibility of demonstrating a task directly, including interaction forces, without relying on prior programming. It remains as future work to combine the functionalities in Chapter 7 and this chapter into one program, allowing for an operator to demonstrate contact forces, which can then be learned and subsequently achieved autonomously by the robot. Further, it would be straightforward to use this interface to demonstrate movement and represent it as DMPs. In the context of DMPs in contact operations, an interesting direction of research considers compliant DMPs, presented in [Denisa et al., 2016; Batinica et al., 2017].

8.8 Conclusion

In this chapter, an immersive haptic interface for task demonstration using two manipulators has been described. It allows for an operator to demonstrate both movements and interaction forces in an intuitive way. The approach is based on introducing virtual constraints between the manipulators, ensuring that the offset between the end-effectors remains constant. Based on this, a control law coupling the motion of two nonlinear robotic arms in task space has been derived. The interface supports using dissimilar arms, mounted arbitrarily and starting in any configurations independently. Interaction forces from the environment, as well as internal limitations such as singularities, are naturally mediated to the operator as haptic feedback. A video illustrating the interface is available online [Ghazaei Ardakani et al., 2019].

Appendix A. Proof of Kinematic Constraint

In this appendix, a proof of Theorem 8.1 is presented. It was originally presented in [Ghazaei Ardakani et al., 2018] where it had been formulated by the first author, and is included here as well for the sake of completeness.

Consider first the orientation, and note that for a rotation matrix R the inverse exists and is given by R^{T}. Then the following relation holds.

$$R_1^{\mathrm{T}} R_2 = \Delta R \Leftrightarrow R_2 = R_1 \Delta R \tag{8.13}$$

Differentiating with respect to time yields

$$S(\omega_2) R_2 = S(\omega_1) R_1 \Delta R = S(\omega_1) R_2 \tag{8.14}$$

Here $S(\omega)$ is the skew-symmetric matrix corresponding to the vector product by the angular velocity ω, as explained in [Spong et al., 2006]. Hence,

$$S(\omega_2 - \omega_1) R_2 = 0 \tag{8.15}$$

and multiplication with R_2^{T} on both sides yields

$$S(\omega_2 - \omega_1) = 0 \Rightarrow \omega_2 - \omega_1 = 0 \tag{8.16}$$

Further, differentiating the position constraint in (8.1) yields $\dot{p}_2 - \dot{p}_1$, and hence the kinematic constraint is

$$\dot{p}_2 - \dot{p}_1 = 0 \tag{8.17a}$$

$$\omega_2 - \omega_1 = 0 \tag{8.17b}$$

Rewriting this using Jacobians and joint velocities yields

$$J_2(q_2)\dot{q}_2 - J_1(q_1)\dot{q}_1 = 0 \tag{8.18}$$

or, equivalently,

$$G\dot{q} = 0 \tag{8.19}$$

This completes the proof.

9 Conclusion and Future Research

This thesis presented research in the context of human–robot interaction. It provided methods based on motion control, force control, and machine learning to make robot programming more accessible, while enhancing replanning capabilities of robots. Each of the proposed methods was verified through experiments on an industrial collaborative robot.

Robot programming by demonstration has been a theme throughout the thesis. Lead-through programming was used to demonstrate desired movements, and to modify faulty movements in existing programs thus saving time as compared to conventional programming. Temporal coupling for DMPs was made realizable, allowing for robots to recover from unforeseen disturbances and detours during task execution.

The proposed algorithms also enable demonstration of task-specific forces. A haptic interface was developed, allowing for an operator to demonstrate contact forces remotely, while experiencing the forces through feedback. Machine learning was used to recognize force transients based on example data, to enable autonomous monitoring of manipulation tasks.

The proposed functionalities could be combined in future products, even though they were evaluated separately in this thesis. For instance, the haptic interface could be used to demonstrate movement and forces for a given task. A DMP could then be used to execute the movement, and force measurements could be used to monitor the progress and verify completion.

Even though this thesis mainly described applications for collaborative robot arms, the insights could be used for control of other machines such as mobile robots and unmanned aerial vehicles. For instance, DMPs were used for control of a humanoid in [Ijspeert et al., 2002], a helicopter process in [Perk and Slotine, 2006], and a legged robot in [Wensing and Slotine, 2017].

This thesis was limited to cases where the setting during the autonomous operation was similar to that of the demonstration. An ability of robots to accomplish general manipulation tasks, without prior programming or demonstration specific for a certain task, is still very distant. Because it is not feasible to manually specify rules for general manipulation, machine learning will probably continue to be an important field while approaching this.

An interesting direction of future work would be to use demonstrated programs to warm-start reinforcement learning, for instance to accomplish assembly tasks. In such a setting, a reward is typically assigned to the robot once the task is successfully finished. Detection of task completion can be done manually, or through manually specified criteria. In this thesis, autonomous detection of task completion, learned from previous demonstrations, was developed. Such detection may be used to determine when to assign rewards in reinforcement learning.

Further, DMPs could represent prior information, and exploration could be achieved through small deviations from the preliminary movement. During the learning procedure, it would be likely that the assembly took more time to accomplish as compared to a demonstrated movement. It would then be useful to slow down the DMP evolution rate through temporal coupling. Reinforcement learning and earlier DMP versions have been combined in, *e.g.*, [Stulp et al., 2012a; Stulp et al., 2012b; Li et al., 2018]. Further, the reinforcement learning in [Levine and Koltun, 2013; Levine et al., 2015] is a promising approach to achieve robotic assembly, though impractical amounts of trials are required if prior information is not included. It would therefore be valuable to explore combinations of these previous methods and the methods proposed in this thesis.

Bibliography

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. (2016). "TensorFlow: large-scale machine learning on heterogeneous distributed systems". *arXiv:1603.04467*.

ABB Robotics (2018). *YuMi — IRB 14000*. URL: `http://new.abb.com/products/robotics/yumi` (visited on 2019-02-28).

ABB Robotics (2019a). *ABB IRC5*. URL: `http://new.abb.com/products/robotics/controllers/irc5` (visited on 2019-02-28).

ABB Robotics (2019b). *ABB RobotStudio*. URL: `https://new.abb.com/products/robotics/robotstudio/` (visited on 2019-03-02).

ABB Robotics (2019c). *Application Manual — Controller Software IRC5*. URL: `https://us.v-cdn.net/5020483/uploads/editor/7y/p0a51v5nb8kg.pdf` (visited on 2019-02-06).

Abu-Dakka, F. J., B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude (2015). "Adaptation of manipulation skills in physical contact with the environment to reference force profiles". *Autonomous Robots* **39**:2, pp. 199–217.

Aliaga, I., A. Rubio, and E. Sanchez (2004). "Experimental quantitative comparison of different control architectures for master-slave teleoperation". *IEEE Transactions on Control Systems Technology* **12**:1, pp. 2–11.

Anderson, R. and M. Spong (1989). "Bilateral control of teleoperators with time delay". *IEEE Transactions on Automatic Control* **34**:5, pp. 494–501.

Argall, B. D., S. Chernova, M. Veloso, and B. Browning (2009). "A survey of robot learning from demonstration". *Robotics and Autonomous Systems* **57**:5, pp. 469–483.

Åström, K. J. and B. Wittenmark (2013). *Computer-Controlled Systems: Theory and Design*. Courier Corporation, Mineola, NY.

Åström, K. J. and T. Hägglund (1995). *PID Controllers: Theory, Design, and Tuning*. Instrument Society of Automation, Research Triangle Park, NC.

ATI (2019). *ATI: Industrial Automation*. URL: `https://www.ati-ia.com/` (visited on 2019-03-03).

Atkeson, C. G., A. W. Moore, and S. Schaal (1997). "Locally weighted learning for control". In: *Lazy Learning*. Springer, Dordrecht, Netherlands, pp. 75–113.

Bagge Carlson, F. (2016). *DynamicMovementPrimitives.jl*. Dept. Automatic Control, Lund University. URL: `https://github.com/baggepinnen/DynamicMovementPrimitives.jl` (visited on 2019-02-28).

Bagge Carlson, F. (2019). *Machine Learning and System Identification for Estimation in Physical Systems*. PhD thesis. TFRT–1122–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Bagge Carlson, F. and M. Haage (2017). "YuMi low-level motion guidance using the Julia programming language and externally guided motion research interface". *Technical Reports* TFRT-7651. Dept. Automatic Control, Lund University, Lund, Sweden.

Bagge Carlson, F., M. Karlsson, A. Robertsson, and R. Johansson (2016). "Particle filter framework for 6D seam tracking under large external forces using 2D laser sensors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. October 9–14, Daejeon, South Korea, pp. 3728–3734.

Bagge Carlson, F., A. Robertsson, and R. Johansson (2015). "Modeling and identification of position and temperature dependent friction phenomena without temperature sensing". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 28–October 2, Hamburg, Germany, pp. 3045–3051.

Batinica, A., B. Nemec, A. Ude, M. Rakovi, and A. Gams (2017). "Compliant movement primitives in a bimanual setting". *arXiv:1707.04629*.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2014). "Julia: a fresh approach to numerical computing". *arXiv:1411.1607*.

Bhat, S. P. and D. S. Bernstein (2000). "A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon". *Systems & Control Letters* **39**:1, pp. 63–70.

Bishop, C. (2007). *Pattern Recognition and Machine Learning*. Springer, New York.

Björkelund, A., L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Störkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx (2011). "On the integration of skilled robot motions for productivity in manufacturing". In: *IEEE/CIRP International Symposium on Assembly and Manufacturing (ISAM)*. May 25–27, Tampere, Finland.

Blomdell, A., G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, and J. Wang (2005). "Extending an industrial robot controller: implementation and applications of a fast open sensor interface". *IEEE Robotics & Automation Magazine* **12**:3, pp. 85–94.

Blomdell, A., I. Dressler, K. Nilsson, and A. Robertsson (2010). "Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 3–8, Anchorage, Alaska, pp. 62–66.

Boyle, M. (2017). "The integration of angular velocity". *Advances in Applied Clifford Algebras* **27**:3, pp. 2345–2374.

Capurso, M., M. M. Ghazaei Ardakani, R. Johansson, A. Robertsson, and P. Rocco (2017). "Sensorless kinesthetic teaching of robotic manipulators assisted by observer-based force control". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 29–June 3, Singapore, pp. 945–950.

Chen, N., M. Karl, and P. van der Smagt (2016). "Dynamic movement primitives in latent space of time-dependent variational autoencoders". In: *IEEE International Conference on Humanoid Robotics*. IEEE. November 15–17, Cancun, Mexico, pp. 629–636.

Chen, Y., J. Zhang, C. Yang, and B. Niu (2007). "The workspace mapping with deficient-DOF space for the PUMA 560 robot and its exoskeleton arm by using orthogonal experiment design method". *Robotics and Computer-Integrated Manufacturing* **23**:4, pp. 478–487.

Constantinescu, D. (2008). "The feel of virtual mechanisms". In: *Product Engineering*. Springer, Berlin Heidelberg, Germany, pp. 231–242.

Constantinescu, D., S. E. Salcudean, and E. A. Croft (2006). "Haptic manipulation of serial-chain virtual mechanisms". *Journal of Dynamic Systems, Measurement, and Control* **128**:1, pp. 65–74.

Cook, G. E., R. Crawford, D. E. Clark, and A. M. Strauss (2004). "Robotic friction stir welding". *Industrial Robot: An International Journal* **31**:1, pp. 55–63.

Crossley, M. D. (2006). *Essential Topology*. Springer, London, UK.

CVX Research Inc. (2019). *CVX: Matlab software for disciplined convex programming*. URL: `http://cvxr.com/cvx/` (visited on 2019-01-07).

De Backer, J. (2014). *Feedback Control of Robotic Friction Stir Welding*. PhD thesis. University West, Dept. Engineering Science, Trollhättan, Sweden.

De Backer, J. and G. Bolmsjö (2014). “Deflection model for robotic friction stir welding”. *Industrial Robot: An International Journal* **41**:4, pp. 365–372.

Denisa, M., A. Gams, A. Ude, and T. Petri (2016). “Learning compliant movement primitives through demonstration and statistical generalization”. *IEEE/ASME Transactions on Mechatronics* **21**:5, pp. 2581–2594.

Fox, E., M. I. Jordan, E. B. Sudderth, and A. S. Willsky (2009). “Sharing features among dynamical systems with beta processes”. In: *Advances in Neural Information Processing Systems (NIPS)*. December 6–9, Vancouver, Canada, pp. 549–557.

Franka Emika (2018). *Panda*. URL: `https://www.franka.de/panda/` (visited on 2018-10-30).

Freidovich, L., A. Robertsson, A. Shiriaev, and R. Johansson (2008). “Periodic motions of the pendubot via virtual holonomic constraints: theory and experiments”. *Automatica* **44**:3, pp. 785–791.

García, J. G., A. Robertsson, J. G. Ortega, and R. Johansson (2006). “Generalized contact force estimator for a robot manipulator”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 15–19, Orlando, FL, pp. 4019–4024.

Gazebo (2018). *Robot simulation made easy*. URL: `http://gazebosim.org/` (visited on 2018-10-31).

Ghazaei Ardakani, M. M. (2016). *On Trajectory Generation for Robots*. PhD thesis. TFRT–1116–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Ghazaei Ardakani, M. M., M. Karlsson, K. Nilsson, A. Robertsson, and R. Johansson (2018). “Master-slave coordination using virtual constraints for a redundant dual-arm haptic interface”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. October 1–5, Madrid, Spain, pp. 8751–8757.

Ghazaei Ardakani, M. M., M. Karlsson, K. Nilsson, A. Robertsson, and R. Johansson (2019). *Master-slave coordination using virtual constraints for a redundant dual-arm haptic interface*. Dept. Automatic Control, Lund University. URL: `https://www.youtube.com/watch?v=4u57bEA4bqg` (visited on 2019-01-10).

Giszter, S. F., F. A. Mussa-Ivaldi, and E. Bizzi (1993). “Convergent force fields organized in the frog’s spinal cord”. *Journal of Neuroscience* **13**:2, pp. 467–491.

Glad, T. and L. Ljung (2000). *Control Theory*. CRC Press, Miami, FL.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. (visited on 2019-02-28). MIT Press, Cambridge, MA. URL: `http://www.deeplearningbook.org`.

Grant, M. C. and S. P. Boyd (2008). “Graph implementations for nonsmooth convex programs”. In: *Recent Advances in Learning and Control*. Springer, Berlin Heidelberg, Germany, pp. 95–110.

Graves, A. (2012). “Neural networks”. In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, Berlin Heidelberg, Germany, pp. 15–35.

Haage, M., S. Profanter, I. Kessler, A. Perzylo, N. Somani, O. Sörnmo, M. Karlsson, S. G. Robertz, K. Nilsson, L. Resch, and M. Marti (2016). “On cognitive robot woodworking in SMErobotics”. In: *ISR 2016: 47th International Symposium on Robotics*. VDE. June 21–22, Munich, Germany, pp. 1–7.

Hatanaka, T., N. Chopra, and M. W. Spong (2015). “Passivity-based control of robots: historical perspective and contemporary issues”. In: *IEEE Conference on Decision and Control (CDC)*. December 15–18, Osaka, Japan, pp. 2450–2452.

Hatcher, A. (2002). *Algebraic Topology*. Cambridge University Press, Cambridge, England.

Hokayem, P. F. and M. W. Spong (2006). “Bilateral teleoperation: an historical survey”. *Automatica* **42**:12, pp. 2035–2057.

Hripcsak, G. and A. S. Rothschild (2005). “Agreement, the F-measure, and reliability in information retrieval”. *Journal of the American Medical Informatics Association* **12**:3, pp. 296–298.

Huggett, S. and D. Jordan (2009). *A Topological Aperitif*. Springer, London, UK.

Ijspeert, A., J. Nakanishi, and S. Schaal (2003). “Learning control policies for movement imitation and movement recognition”. In: *Neural Information Processing System (NIPS)*. Vol. 15. December 5–10, Stateline, Nevada, pp. 1547–1554.

Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal (2013). “Dynamical movement primitives: learning attractor models for motor behaviors”. *Neural Computation* **25**:2, pp. 328–373.

Ijspeert, A. J., J. Nakanishi, and S. Schaal (2002). “Movement imitation with nonlinear dynamical systems in humanoid robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. May 11–15, Washington, DC, pp. 1398–1403.

Isidori, A. (1995). *Nonlinear Control Systems*. Springer Verlag, London, UK.

Itoh, T., K. Kosuge, and T. Fukuda (2000). “Human-machine cooperative telemanipulation with motion and force scaling using task-oriented virtual tool dynamics”. *IEEE Transactions on Robotics and Automation* **16**:5, pp. 505–516.

Japkowicz, N. (2000). “The class imbalance problem: significance and strategies”. In: *International Conference on Artificial Intelligence*. June 26–29, Las Vegas, Nevada.

Japkowicz, N. and S. Stephen (2002). “The class imbalance problem: a systematic study”. *Intelligent Data Analysis* **6**:5, pp. 429–449.

Johansson, R. (1993). *System Modeling and Identification*. Prentice-Hall, Englewood Cliffs, NJ.

Johansson, R., M. Annerstedt, and A. Robertsson (2009). “Stability of haptic obstacle avoidance and force interaction”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. October 11–15, St. Louis, MO, pp. 3238–3243.

Johansson, R., K. Nilsson, and A. Robertsson (2015). “Force control”. In: *Handbook of Manufacturing Engineering and Technology*. Springer Verlag, London, UK, pp. 1933–1965.

Johansson, R. and A. Robertsson (2009). “Stability of robotic obstacle avoidance and force interaction”. In: *International Symposium on Robot Control (SYROCO’09)*. September 9–12, Gifu, Japan, pp. 561–566.

Kahn, D. W. (1975). *Topology: An Introduction to the Point-Set and Algebraic Areas*. Courier Corporation, North Chelmsford, MA.

Karl, M., M. Soelch, J. Bayer, and P. van der Smagt (2016). “Deep variational Bayes filters: unsupervised learning of state space models from raw data”. *arXiv:1605.06432*.

Karlsson, F., M. Karlsson, B. Bernhardsson, F. Tufvesson, and M. Persson (2015). “Sensor fused indoor positioning using dual band WiFi signal measurements”. In: *European Control Conference (ECC)*. July 15–17, Linz, Austria, pp. 1669–1672.

Karlsson, M. (2016). *Experimental evaluation of DMP perturbation recovery*. Dept. Automatic Control, Lund University. URL: `https://www.youtube.com/watch?v=u8GwsSsL0TI` (visited on 2019-02-28).

Karlsson, M. (2017a). *Convergence of DMPs - experimental evaluation*. Dept. Automatic Control, Lund University. URL: `https://www.youtube.com/watch?v=KRfjMXs4LjQ` (visited on 2019-02-28).

Karlsson, M. (2017b). *Detection of contact force transients from robot joint torques*. Dept. Automatic Control, Lund University. URL: `https://www.youtube.com/watch?v=TE1q5lQr4nk` (visited on 2018-12-18).

Karlsson, M. (2017c). *DMP perturbation, simulation example*. Dept. Automatic Control, Lund University. URL: `https://gitlab.control.lth.se/cont-mkr/dmp_perturbation_sim_example` (visited on 2019-02-28).

Karlsson, M. (2017d). *Modification of dynamical movement primitives*. Dept. Automatic Control, Lund University. URL: `https://www.youtube.com/watch?v=q998JUwofX4&feature=youtu.be` (visited on 2019-02-28).

Karlsson, M. (2017e). *On Motion Control and Machine Learning for Robotic Assembly*. Licentiate Thesis, TFRT–3274–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Karlsson, M. (2019). *Temporally coupled DMPs in Cartesian space*. Dept. Automatic Control, Lund University. URL: `https://www.youtube.com/watch?v=r0UMux3KH64` (visited on 2019-02-06).

Karlsson, M., F. Bagge Carlson, J. De Backer, M. Holmstrand, A. Robertsson, and R. Johansson (2016). "Robotic seam tracking for friction stir welding under large contact forces". In: *7th Swedish Production Symposium (SPS)*. October 25–27, Lund, Sweden.

Karlsson, M., F. Bagge Carlson, A. Robertsson, and R. Johansson (2017a). "Two-degree-of-freedom control for trajectory tracking and perturbation recovery during execution of dynamical movement primitives". In: *20th IFAC World Congress*. July 9–14, Toulouse, France, pp. 1959–1966.

Karlsson, M. and F. Karlsson (2016). "Cooperative indoor positioning by exchange of bluetooth signals and state estimates between users". In: *European Control Conference (ECC)*. June 29–July 1, Aalborg, Denmark, pp. 1440–1444.

Karlsson, M., A. Robertsson, and R. Johansson (2017b). "Autonomous interpretation of demonstrations for modification of dynamical movement primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 29–June 3, Singapore, pp. 316–321.

Karlsson, M., A. Robertsson, and R. Johansson (2018a). "Convergence of dynamical movement primitives with temporal coupling". In: *European Control Conference (ECC)*. June 12–15, Limassol, Cyprus, pp. 32–39.

Karlsson, M., A. Robertsson, and R. Johansson (2018b). "Detection and control of contact force transients in robotic manipulation without a force sensor". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 21–25, Brisbane, Australia, pp. 21–25.

Karlsson, M., A. Robertsson, and R. Johansson (2019). "Temporally coupled dynamical movement primitives in Cartesian space". Manuscript prepared for submission to review for publication.

Keating, J. and W. Thach (1997). "No clock signal in the discharge of neurons in the deep cerebellar nuclei". *Journal of Neurophysiology* **77**:4, pp. 2232–2234.

Khalil, H. K. (2002). *Nonlinear Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Khatib, O. (1986). “Real-time obstacle avoidance for manipulators and mobile robots”. *The International Journal of Robotics Research* **5**:1, pp. 90–98.

Kingma, D. and J. Ba (2014). “Adam: a method for stochastic optimization”. *arXiv:1412.6980*.

Kober, J., B. Mohler, and J. Peters (2008). “Learning perceptual coupling for motor primitives”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 22–26, Nice, France, pp. 834–839.

Kock, S., T. Vittor, B. Matthias, H. Jerregard, M. Källman, I. Lundberg, R. Mellander, and M. Hedelind (2011). “Robot concept for scalable, flexible assembly automation: a technology study on a harmless dual-armed robot”. In: *International Symposium on Assembly and Manufacturing (ISAM)*. IEEE. May 25–27, Tampere, Finland, pp. 1–5.

Koditschek, D. (1987). “Exact robot navigation by means of potential functions: some topological considerations”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. March 31–April 3, Raleigh, NC, pp. 1–6.

Koditschek, D. E. (1988). “Application of a new Lyapunov function to global adaptive attitude tracking”. In: *27th IEEE Conference on Decision and Control*. December 7-9, Austin, TX, pp. 63–68.

Kroemer, O., R. Detry, J. Piater, and P. J (2010). “Combining active learning and reactive control for robot grasping”. *Robotics and Autonomous Systems (RAS)* **58**:9, pp. 1105–1116.

Kröse, B. and P. van der Smagt (1993). *An Introduction to Neural Networks*. Citeseerx, State College, PA.

KUKA (2018). *LBR iiwa*. URL: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa (visited on 2018-10-30).

Kulvicius, T., K. Ning, M. Tamosiunaite, and F. Worgotter (2012). “Joining movement sequences: modified dynamic movement primitives for robotics applications exemplified on handwriting”. *IEEE Transactions on Robotics* **28**:1, pp. 145–157.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, UK.

Lawrence, D. (1993). “Stability and transparency in bilateral teleoperation”. *IEEE Transactions on Robotics and Automation* **9**:5, pp. 624–637.

Lee, D. and M. Spong (2006). “Passive bilateral teleoperation with constant time delay”. *IEEE Transactions on Robotics* **22**:2, pp. 269–281.

Lee, D. and P. Y. Li (2003). "Passive bilateral feedforward control of linear dynamically similar teleoperated manipulators". *IEEE Transactions on Robotics and Automation* **19**:3, pp. 443–456.

Lee, D. and P. Y. Li (2005). "Passive bilateral control and tool dynamics rendering for nonlinear mechanical teleoperators". *IEEE Transactions on Robotics* **21**:5, pp. 936–951.

Lee, S. H., I. H. Suh, S. Calinon, and R. Johansson (2015). "Autonomous framework for segmenting robot trajectories of manipulation task". *Autonomous Robots* **38**:2, pp. 107–141.

Lehner, J. (1964). *Discontinuous Groups and Automorphic Functions*. American Mathematical Society, Providence, RI.

Levine, S. and V. Koltun (2013). "Guided policy search". In: *International Conference on Machine Learning (ICML)*. June 16–21, Atlanta, GA, pp. 1–9.

Levine, S., N. Wagener, and P. Abbeel (2015). "Learning contact-rich manipulation skills with guided policy search". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 26–30, Seattle, WA, pp. 156–163.

Levy, A. (2002). *Basic Set Theory*. Dover Publications, Mineola, NY.

Li, P. Y. and R. Horowitz (1999). "Passive velocity field control of mechanical manipulators". *IEEE Transactions on Robotics and Automation* **15**:4, pp. 751–763.

Li, Z., T. Zhao, F. Chen, Y. Hu, C.-Y. Su, and T. Fukuda (2018). "Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator". *IEEE/ASME Transactions on Mechatronics* **23**:1, pp. 121–131.

Linderoth, M. (2013). *On Robotic Work-Space Sensing and Control*. PhD thesis. TFRT–1098–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Linderoth, M., A. Stolt, A. Robertsson, and R. Johansson (2013). "Robotic force estimation using motor torques and modeling of low velocity friction disturbances". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. November 3–7, Tokyo, Japan, pp. 3550–3556.

Lindsten, F., T. B. Schön, and M. I. Jordan (2013). "Bayesian semiparametric Wiener system identification". *Automatica* **49**:7, pp. 2053–2063.

Ljung, L. (1987). *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ.

Lohmiller, W. and J. J. E. Slotine (1998). "On contraction analysis for nonlinear systems". *Automatica* **34**:6, pp. 683–696.

Malmaud, J. (2017). *A Julia wrapper for TensorFlow*. URL: `https://github.com/malmaud/TensorFlow.jl` (visited on 2019-02-28).

Marchal, P. C., O. Sörnmo, B. Olofsson, A. Robertsson, J. G. Ortega, and R. Johansson (2014). "Iterative learning control for machining with industrial robots". In: *19th IFAC World Congress*. August 24–29, Cape Town, South Africa, pp. 9327–9333.

Marzinotto, A., M. Colledanchise, C. Smith, and P. Ögren (2014). "Towards a unified behavior trees framework for robot control". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 31–June 7, Hong Kong, China, pp. 5420–5427.

Matsubara, T., S.-H. Hyon, and J. Morimoto (2011). "Learning parametric dynamic movement primitives from multiple demonstrations". *Neural Networks* **24**:5, pp. 493–500.

Matthias, B., S. Kock, H. Jerregard, M. Källman, and I. Lundberg (2011). "Safety of collaborative industrial robots: certification possibilities for a collaborative assembly robot concept". In: *International Symposium on Assembly and Manufacturing (ISAM)*. IEEE. May 25–27, Tampere, Finland, pp. 1–6.

Mattingley, J. and S. Boyd (2012). "CVXGEN: a code generator for embedded convex optimization". *Optimization and Engineering* **13**:1, pp. 1–27.

Mayhew, C. G., R. G. Sanfelice, and A. R. Teel (2011). "Quaternion-based hybrid control for robust global attitude tracking". *IEEE Transactions on Automatic Control* **56**:11, pp. 2555–2566.

Meier, F., E. Theodorou, F. Stulp, and S. Schaal (2011). "Movement segmentation using a primitive library". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 25–30, San Francisco, CA, pp. 3407–3412.

Miyamoto, H., S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato (1996). "A Kendama learning robot based on bi-directional theory". *Neural Networks* **9**:8, pp. 1281–1302.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press, Cambridge, MA.

Mussa-Ivaldi, F. A. (1999). "Modular features of motor control and learning". *Current Opinion in Neurobiology* **9**:6, pp. 713–717.

Nemec, B., L. lajpah, S. lajpa, J. Pikur, and A. Ude (2018). "An efficient PbD framework for fast deployment of bi-manual assembly tasks". In: *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. November 6–9, Beijing, China, pp. 166–173.

Niekum, S., S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto (2015). “Learning grounded finite-state representations from unstructured demonstrations”. *The International Journal of Robotics Research* **34**:2, pp. 131–157.

Norrlöf, M. (2002). “An adaptive iterative learning control algorithm with experiments on an industrial robot”. *IEEE Transactions on Robotics and Automation* **18**:2, pp. 245–251.

Norrlöf, M. and S. Gunnarsson (2002). “Experimental comparison of some classical iterative learning control algorithms”. *IEEE Transactions on Robotics and Automation* **18**:4, pp. 636–641.

Ögren, P. (2012). “Increasing modularity of UAV control systems using computer game behavior trees”. In: *AIAA Guidance, Navigation, and Control Conference*. August 13–16, Minneapolis, Minnesota, p. 4458.

Olofsson, B. (2015). *Topics in Machining with Industrial Robot Manipulators and Optimal Motion Control*. PhD thesis. TFRT–1108–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Olsson, T., J. Bengtsson, R. Johansson, and H. Malm (2002). “Force control and visual servoing using planar surface identification”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 11–15, Washington, DC, pp. 4211–4216.

Pan, Z., J. Polden, N. Larkin, S. Van Duin, and J. Norrish (2010). “Recent progress on programming methods for industrial robots”. In: *41st International Symposium on Robotics (ISR)*. VDE. June 7–9, Munich, Germany, pp. 1–8.

Panchapagesan, S., M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni (2016). “Multi-task learning and weighted cross-entropy for DNN-based keyword spotting”. *Interspeech 2016*, pp. 760–764.

Papageorgiou, D., A. Sidiropoulos, and Z. Doulgeri (2018). “Sinc-based dynamic movement primitives for encoding point-to-point kinematic behaviors”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. October 1–5, Madrid, Spain, pp. 8339–8345.

Park, D.-H., H. Hoffmann, P. Pastor, and S. Schaal (2008). “Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields”. In: *Humanoids 2008—8th IEEE-RAS International Conference on Humanoid Robots*. December 1–3, Daejeon, South Korea, pp. 91–98.

Pastor, P., H. Hoffmann, T. Asfour, and S. Schaal (2009). “Learning and generalization of motor skills by learning from demonstration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 12–17, Kobe, Japan, pp. 763–768.

Pastor, P., M. Kalakrishnan, F. Meier, F. Stulp, J. Buchli, E. Theodorou, and S. Schaal (2013). “From dynamic movement primitives to associative skill memories”. *Robotics and Autonomous Systems* **61**:4, pp. 351–361.

Perk, B. E. and J. J. E. Slotine (2006). “Motion primitives for robotic flight control”. *arXiv:cs/0609140*.

Persson, M., M. Karlsson, and F. Karlsson (2014). *Method for improved indoor positioning and crowd sourcing using PDR*. International patent, Publication number: WO 2015/118369 (A1), Applicant: Sony Corporation, Examined and granted under the patent cooperation treaty (PCT). Sony. URL: https://patents.google.com/patent/WO2015118369A1/en (visited on 2019-03-02).

Prada, M., A. Remazeilles, A. Koene, and S. Endo (2014). “Implementation and experimental validation of dynamic movement primitives for object handover”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 14–18, Chicago, Illinois, pp. 2146–2153.

Rebelo, J. and A. Schiele (2012). “Master-slave mapping and slave base placement optimization for intuitive and kinematically robust direct teleoperation”. In: *International Conference on Control, Automation and Systems (ICCAS)*. October 17–21, Jeju Island, South Korea, pp. 2017–2022.

Rethink Robotics (2018). *Baxter*. URL: https://www.rethinkrobotics.com/baxter/ (visited on 2018-10-30).

Rodriguez, A., D. Bourne, M. Mason, G. F. Rossano, and J. Wang (2010). “Failure detection in assembly: force signature analysis”. In: *IEEE Conference on Automation Science and Engineering (CASE)*. August 21–24, Toronto, Canada, pp. 210–215.

Rojas, J., K. Harada, H. Onda, N. Yamanobe, E. Yoshida, K. Nagata, and Y. Kawai (2012). “A relative-change-based hierarchical taxonomy for cantilever-snap assembly verification”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. October 7–12, Vilamoura, Portugal, pp. 356–363.

ROS (2018). *rviz Package Summary*. URL: http://wiki.ros.org/rviz (visited on 2018-10-31).

Rubinstein, R. Y. and D. P. Kroese (2013). *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Science & Business Media, New York.

Rugh, W. J. (1996). *Linear System Theory*. Vol. 2. Prentice-Hall, Upper Saddle River, NJ.

Sanderson, C. and R. Curtin (2016). “Armadillo: a template-based C++ library for linear algebra”. *Journal of Open Source Software* **1**, p. 26.

SARAFun (2019). *Horizon 2020 SARAFun — Smart Assembly Robot with Advanced Functionalities*. URL: `http://h2020sarafun.eu/` (visited on 2019-01-07).

Schaal, S. and C. G. Atkeson (1998). “Constructive incremental learning from only local information”. *Neural Computation* **10**:8, pp. 2047–2084.

Schaal, S., A. Ijspeert, and A. Billard (2003). “Computational approaches to motor learning by imitation”. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* **358**:1431, pp. 537–547.

Schaal, S., S. Kotosaka, and D. Sternad (2000). “Nonlinear dynamical systems as movement primitives”. In: *IEEE International Conference on Humanoid Robotics*. September 7–8, Boston, MA, pp. 1–11.

Schön, T. B., F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson, and L. Dai (2015). “Sequential Monte Carlo methods for system identification”. In: *IFAC Symposium on System Identification (SYSID)*. October 19–21, Beijing, China, pp. 775–786.

Schwarz, A. S. (2013). *Topology for Physicists*. Vol. 308. Springer, Berlin Heidelberg, Germany.

Shiriaev, A. S., L. B. Freidovich, A. Robertsson, R. Johansson, and A. Sandberg (2007). “Virtual-holonomic-constraints-based design of stable oscillations of Furuta pendulum: theory and experiments”. *IEEE Transactions on Robotics* **23**:4, pp. 827–832.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2010). *Robotics: Modelling, Planning and Control*. Springer Verlag, London, UK.

Slotine, J. J. E. and W. Li (1991). *Applied Nonlinear Control*. Vol. 199. 1. Prentice-Hall, Englewood Cliffs, NJ.

Spong, M. W., S. Hutchinson, and M. Vidyasagar (2006). *Robot Modeling and Control*. John Wiley & Sons, Hoboken, NJ.

Stavridis, S., D. Papageorgiou, and Z. Doulgeri (2017). “Dynamical system based robotic motion generation with obstacle avoidance”. *Robotics and Automation Letters* **2**:2, pp. 712–718.

Stolt, A. (2015). *On Robotic Assembly using Contact Force Control and Estimation*. PhD thesis. TFRT–1109–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Stolt, A., F. B. Carlson, M. M. Ghazaei Ardakani, I. Lundberg, A. Robertsson, and R. Johansson (2015a). “Sensorless friction-compensated passive lead-through programming for industrial robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 28–October 2, Hamburg, Germany, pp. 3530–3537.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2011). “Force controlled assembly of emergency stop button”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 9–13, Shanghai, China, pp. 3751–3756.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012a). “Adaptation of force control parameters in robotic assembly”. In: *10th IFAC Symposium on Robot Control*. September 5–7, Dubrovnik, Croatia, pp. 561–566.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012b). “Force controlled robotic assembly without a force sensor”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 14–18, St. Paul, MN, pp. 1538–1543.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2015b). “Detection of contact force transients in robotic assembly”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 26–30, Seattle, WA, pp. 962–968.

Stolt, A., A. Robertsson, and R. Johansson (2015c). “Robotic force estimation using dithering to decrease the low velocity friction uncertainties”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 26–30, Seattle, WA, pp. 3896–3902.

Stulp, F., J. Buchli, A. Ellmer, M. Mistry, E. A. Theodorou, and S. Schaal (2012a). “Model-free reinforcement learning of impedance control in stochastic environments”. *IEEE Transactions on Autonomous Mental Development* **4**:4, pp. 330–341.

Stulp, F. and S. Schaal (2011). “Hierarchical reinforcement learning with movement primitives”. In: *IEEE-RAS International Conference on Humanoid Robots*. October 26–28, Bled, Slovenia, pp. 231–238.

Stulp, F., E. A. Theodorou, and S. Schaal (2012b). “Reinforcement learning with sequences of motion primitives for robust manipulation”. *IEEE Transactions on Robotics* **28**:6, pp. 1360–1370.

Stulp, F., E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal (2011). “Learning motion primitive goals for robust manipulation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 25–30, San Francisco, CA, pp. 325–331.

Talignani Landi, C., F. Ferraguti, C. Fantuzzi, and C. Secchi (2018a). “A passivity-based strategy for coaching in human–robot interaction”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 21–25, Brisbane, Australia, pp. 3279–3284.

Talignani Landi, C., F. Ferraguti, C. Fantuzzi, and C. Secchi (2018b). *A passivity-based strategy for coaching in human–robot interaction*. University of Modena and Reggio Emilia, ICRA Spotlight Video. URL: https://www.youtube.com/watch?reload=9&v=ouEY7nrHAH4 (visited on 2018-12-16).

TensorFlow (2019). *An open-source software library for machine intelligence*. URL: https://www.tensorflow.org/ (visited on 2019-02-28).

Theorin, A. (2014). *A Sequential Control Language for Industrial Automation*. PhD thesis. TFRT–1104–SE, Dept. Automatic Control, Lund University, Lund, Sweden.

Thomas, U., S. Molkenstruck, R. Iser, and F. M. Wahl (2007). "Multi sensor fusion in robot assembly using particle filters". In: *IEEE International Conference on Robotics and Automation (ICRA)*. April 10–14, Roma, Italy, pp. 3837–3843.

Thomas, U. and F. M. Wahl (2001). "A system for automatic planning, evaluation and execution of assembly sequences for industrial robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. October 29–November 3, Maui, HI, pp. 1458–1464.

Turro, N., O. Khatib, and E. Coste-Maniere (2001). "Haptically augmented teleoperation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 21–26, Seoul, South Korea, pp. 386–392.

Ude, A. (1999). "Filtering in a unit quaternion space for model-based object tracking". *Robotics and Autonomous Systems* **28**:2-3, pp. 163–172.

Ude, A., B. Nemec, T. Petri, and J. Morimoto (2014). "Orientation in Cartesian space dynamic movement primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 31–June 7, Hong Kong, China, pp. 2997–3004.

Wada, Y. and M. Kawato (2004). "A via-point time optimization algorithm for complex sequential trajectory formation". *Neural Networks* **17**:3, pp. 353–364.

Wadenbäck, M., M. Karlsson, A. Heyden, A. Robertsson, and R. Johansson (2017). "Visual odometry from two point correspondences and initial automatic camera tilt calibration". In: *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Volume 6*. VISIGRAPP. February 27–March 1, Porto, Portugal, pp. 340–346.

Wahrburg, A., S. Klose, D. Clever, T. Groth, S. Moberg, J. Styrud, and H. Ding (2018). "Modeling speed-, load-, and position-dependent friction effects in strain wave gears". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 21–25, Brisbane, Australia, pp. 2095–2102.

Wahrburg, A., S. Zeiss, B. Matthias, and H. Ding (2014). “Contact force estimation for robotic assembly using motor torques”. In: *IEEE International Conference on Automation Science and Engineering (CASE)*. August 18–22, Taipei, Taiwan, pp. 1252–1257.

Wen, J. T. and S. H. Murphy (1990). *PID Control for Robot Manipulators*. Rensselaer Polytechnic Institute.

Wen, J. T. and S. H. Murphy (1991). “Stability analysis of position and force control for robot arms”. *IEEE Transactions on Automatic Control* **36**:3, pp. 365–371.

Wensing, P. M. and J. J. E. Slotine (2017). “Sparse control for dynamic movement primitives”. In: *The 20th World Congress of the International Federation of Automatic Control (IFAC)*. July 9-14, Toulouse, France, pp. 10531–10538.

Westervelt, E. R., J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris (2007). *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, Boca Raton, FL.

Yang, D., Q. Lv, G. Liao, K. Zheng, J. Luo, and B. Wei (2018). “Learning from demonstration: dynamical movement primitives based reusable suturing skill modelling method”. In: *Chinese Automation Congress (CAC)*. IEEE. November 30–December 2, Xi’an, China, pp. 4252–4257.